ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

TESIS DE GRADO

"DISEÑO E IMPLEMENTACION DE UN ROBOT MOVIL TELEOPERADO EN BASE AL RECONOCIMIENTO DE FORMA Y MOVIMIENTO DE OBJETOS"

Previa a la obtención del título de:

INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL
INGENIERO EN COMPUTACIÓN ESPECIALIZACIÓN SISTEMAS MULTIMEDIA
INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

PRESENTADA POR:

MICHAEL DANIEL ERAZO ALVAREZ
DANILO XAVIER MATAMOROS MORALES
JAIME MOISES MINCHALA MARQUINO

GUAYAQUIL – ECUADOR

2007

AGRADECIMIENTO

A todos quienes
contribuyeron en el
desarrollo de este trabajo

DEDICATORIA

A nuestros padres

TRIBUNAL DE GRADUACIÓN

PRESIDENTE

Ing. Holger Cevallos Ulloa

DIRECTOR DE TESIS

Ing. Dennys Paillacho Chiluiza

MIEMBROS PRINCIPALES

Ph.D. Boris Vintimilla

Ing. Carlos Valdivieso

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____
Michael Daniel Erazo Alvarez

_____
Danilo Xavier Matamoros Morales

_____
Jaime Moises Minchala Marquino

RESUMEN

El presente proyecto tiene como finalidad la construcción de un robot móvil que será teleoperado por una aplicación. Esta aplicación utilizará técnicas de procesamiento digital de imágenes y visión por computador para reconocer y seguir objetos tangibles que serán manipulados por el usuario. Los objetos físicos que el usuario manipulará serán figuras geométricas de distintos colores, cada una de ellas representará un comando de control para el robot. La aplicación detectará el movimiento de estos objetos utilizando una cámara de video. Luego, enviará la orden al robot de acuerdo al movimiento y objeto detectado.

Para poder realizar lo anterior se implementará un algoritmo de reconocimiento y seguimiento de objetos de mayor precisión. El mismo que reducirá el ruido presente comúnmente en este tipo de aplicaciones disminuyendo el envió de comandos al robot basándose en información errónea. Dicho algoritmo será de gran utilidad para otras aplicaciones de visión por computador como lo es el caso del futbol robótico.

Para el control del robot se desarrollarán e implementarán algoritmos en un microcontrolador PIC. El cual se encargará de recibir los comandos enviados por la aplicación y gobernar los motores y dispositivos periféricos del robot.

Para mejorar la precisión de control en lazo abierto se usan motores de pasos. De esta manera se contribuirá con el diseño de un robot de mayor precisión para futuras investigaciones en el área de robótica.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Actualmente la forma más común de controlar a los robots es por medio de paneles de control con botones, perillas, etc. Sin embargo, la necesidad de tener una comunicación más directa entre el humano y el robot, está llevando al desarrollo de alternativas que incluyen comandos por voz y reconocimiento de imágenes.

No siempre es necesario que el usuario envíe órdenes al robot de manera directa. En muchos casos se requiere que este responda a acciones realizadas en el mundo real. Un ejemplo de esto es el fútbol robótico en donde las acciones que el robot efectúa dependen de la posición del balón o de los demás jugadores y no de comandos enviados por el usuario.

Este proyecto tiene como objetivo general desarrollar un robot móvil teleoperado mediante el reconocimiento y seguimiento de objetos tangibles.

Entre los objetivos específicos tenemos:

- Implementar un algoritmo de mayor precisión para el reconocimiento y seguimiento de objetos tangibles.

- Implementar un algoritmo que convierta el movimiento de objetos en comandos de control para el robot.
- Diseñar e implementar un robot móvil que interprete adecuadamente los comandos de control enviados por la aplicación.
- Establecer una comunicación vía RF entre la aplicación y el robot.

En el capítulo 1 se identifica y se define el problema a tratar. Además se enumeran los antecedentes, causales y efectos del problema. Al final de este capítulo detallamos el alcance que tendrá la solución planteada.

En el capítulo 2 se presentan las herramientas disponibles que tenemos para la solución del problema. Se mencionan sus principales características y sus ventajas para el caso particular de nuestro proyecto.

En el capítulo 3 se plantea el diseño de la solución al problema mencionado. El cual se divide en tres partes principales: la primera trata sobre el diseño general, la segunda sobre el diseño de la aplicación y la tercera sobre el diseño de los componentes del hardware.

El capítulo 4 se dedica a detallar como se implementó la solución propuesta. Se mencionan las herramientas seleccionadas tanto de software como

hardware. Y se detalla el proceso de implementación de la aplicación y del robot.

En el capítulo 5 se incluyen las pruebas realizadas. Se explica la metodología seguida para realizar cada una de ellas. Además se muestran los resultados experimentales obtenidos.

CAPÍTULO 1

1 ANÁLISIS DEL PROBLEMA

En el presente capítulo se realiza un análisis del problema a solucionar en este proyecto. Se presentan sus antecedentes, causales y efectos. Por último, se detalla el alcance que tendrá nuestra solución al problema descrito.

1.1 Definición del Problema

Muchas aplicaciones requieren que los robots respondan a acciones que se realizan en el mundo real, mas no a comandos enviados por el usuario a través de algún panel de control. En dichas aplicaciones se utiliza la visión por computador para reconocer las acciones o movimientos que ocurren. Las cuales son interpretadas por un programa que envía los comandos al robot.

La necesidad de este tipo de aplicaciones surgen debido al incremento en la complejidad de las acciones y tareas que puede realizar un robot, además de tener mayor precisión al llevar a cabo dichas acciones. Debido a esto, las órdenes que el usuario necesita enviar al robot se tornan también más complejas, lo que hace que ya no sea suficiente el uso de los medios tradicionales como teclado, palanca de mando o ratón para poderlos controlar.

Si aumentan las funcionalidades de un robot, el usuario tendrá mayor número de opciones que puede controlar. Por este motivo es necesario para cierto tipo de aplicaciones incluir medios audiovisuales para mejorar la comunicación y el control. De esta manera se lo realiza de una forma más directa y natural.

El problema que busca solucionar este proyecto es precisamente la limitación que existe en la forma de controlar a un robot mediante una aplicación que use el teclado o el ratón para ingresar los comandos. Para contribuir a la mejora del problema se diseñará e implementará un prototipo que incluye un robot móvil y una aplicación de PC que utilizará técnicas de procesamiento digital de imágenes y visión por computador para reconocer y seguir objetos tangibles. Los cuales serán manipulados por el usuario para dar las órdenes al robot.

Esta investigación también contribuirá a la implementación de alternativas en la forma en que interactúan los seres humanos y las computadoras. Al manipular objetos tangibles para enviar las órdenes al robot, se están aplicando conceptos de “Bits Tangibles”. El cual tiene como finalidad unir el vacío que existe entre el mundo de los bits y el ambiente físico [1].

Las aplicaciones que se le pueden dar a sistemas de este tipo son muy diversas. Una de ellas son los modelos del aprendizaje del comportamiento de objetos, el cual puede ser usado para que el computador o un robot interactúe con los humanos. Incluso puede darle más libertad a la aplicación a tomar decisiones basándose en el comportamiento del objeto analizado [2].

Otro ejemplo de aplicación es el reconocimiento de acciones usando el contexto. La cual hará que la comunicación con las computadoras mejore. Entre las aplicaciones más importantes tenemos: Vigilancia automática, interfaces de usuario perceptivas, sistemas de monitoreo y asistencia para discapacitados, robots móviles autónomos, casas inteligentes, entre otras [3].

1.2 Antecedentes, causales y efectos del problema

La robótica, desde sus inicios, en la década de los setenta, ha evolucionado a un ritmo similar al de la tecnología que la sustenta, especialmente en lo que se refiere a la informática. La informática ha permitido un gran desarrollo en el sector industrial, en el control de máquinas y equipos en automatización, y en la robótica industrial en general.

En muchos casos se necesita la intervención del humano para controlar al robot, debido a la falta de inteligencia de estos. En el sector de servicios y robots personales es probablemente donde se necesita una mayor inteligencia del robot, puesto que debe operar en entornos menos definidos y estructurados. En el campo de la interacción y cooperación de la persona con el robot, aún es necesario desarrollar sistemas suficientemente inteligentes que permitan conseguir una interacción persona-robot eficiente [4].

Desde los primeros proyectos en el sector de servicios, que se centraban en el campo asistencial o en el mantenimiento, el progreso ha sido lento y pocas veces la experimentación se ha llevado a cabo

en condiciones de entorno reales. Esto se da porque en muchas aplicaciones de robótica existe la necesidad de percibir el entorno para navegar o para realizar el trabajo previsto. La visión por computador, que es el medio de percepción más importante, aún no ha desarrollado una capacidad de interpretación de escenas suficiente para dotar al robot de la información adecuada para actuar inteligentemente en entornos de operación naturales [4].

Existen limitaciones en el caso de los robots teleoperados, cuando se requiere un humano para el control, o cuando este interactúa con el robot, como ocurre en el campo asistencial. Por lo tanto existe la necesidad de desarrollar interfaces inteligentes que permitan efectuar un control cooperativo, al tiempo que se ofrece una comunicación intuitiva y amigable [4].

Con respecto a la forma como nos comunicamos con las máquinas, a inicios de la década de los ochenta, la estación de trabajo Xerox Star fue la primera generación de GUI (interfaz de usuario gráfica). Este fue el primer sistema comercial que explotaba el uso del ratón y ventanas para la interacción con el usuario. Luego llegaría la Apple Macintosh que crearía una nueva ola en la industria de las

computadoras personales. Actualmente, debido a la popularidad de Microsoft Windows, la GUI es ampliamente usada [1].

Hoy en día las interacciones entre las personas y la información digital están limitadas a las tradicionales interfaces de usuario gráficas (GUI). Las interacciones entre el computador y el usuario se realizan a través de estas GUIs, las cuales son parte del mundo digital, pero no del mundo físico en el cual nosotros vivimos.

Aunque el ser humano ha desarrollado varias habilidades para procesar información a través de interacciones táctiles con objetos físicos, actualmente la forma en que se interactúa con las computadoras no hace uso de estas habilidades. Esto se debe a la falta de diversidad de medios de entrada/salida y a la polarización existente hacia las GUIs [1].

En consecuencia, no estamos aprovechando nuestras habilidades naturales para comunicarnos con el computador. Estamos limitándonos al uso de las GUIs tradicionales, el teclado y el ratón. Esto produce que haya una división entre el mundo físico y el de la

información digital. Haciendo que la comunicación entre los humanos y las máquinas sea limitada y no muy eficiente.

1.3 Alcance del proyecto

Este proyecto consiste en el diseño e implementación de un robot móvil y una aplicación de PC que identifica y sigue el movimiento de objetos predeterminados. Esta aplicación sirve como interfaz de usuario para el control del robot móvil. Además se implementará un circuito que transmitirá los comandos del computador hacia el robot.

En la parte del hardware, el proyecto abarca la construcción del robot en su totalidad. El cual tendrá dos ruedas principales, las cuales serán gobernadas por dos motores independientes.

La aplicación de la PC reconocerá objetos previamente definidos, o que el usuario los podrá definir cuando desee. El reconocimiento de los objetos se basa en su color. Debido a esto, es necesario que la luz ambiental sea controlada y que no varíe. La principal ventaja de usar el color para reconocer un objeto es que la alteración de la forma original del objeto producida por la manipulación del usuario, no afectará su reconocimiento por parte de la aplicación.

La comunicación establecida entre el computador que corre la aplicación y el robot móvil controlado es de una vía. Un circuito transmisor se encarga de enviar los datos desde la PC hacia el robot. El sistema no cuenta con retroalimentación del estado del robot hacia la PC.

CAPÍTULO 2

2 ANÁLISIS DE LAS HERRAMIENTAS DISPONIBLES

En el presente capítulo se introducen los conceptos de las herramientas utilizadas en la realización de este proyecto. En principio describiremos las técnicas de procesamiento digital de imágenes y visión por computador. Luego continuaremos con las herramientas utilizadas para la transmisión de datos de la aplicación al robot. Por último se detallará el hardware utilizado en la construcción del robot móvil.

2.1 Proceso de adquisición de imágenes

El proceso de adquisición de imágenes es el primer paso en toda aplicación de visión por computador. La adquisición se la puede realizar con distintos dispositivos. Los más comunes son: cámaras de video, escáneres y cámaras fotográficas.

El funcionamiento de la mayoría de ellos está basado en un sensor de imágenes llamado CCD (del inglés *charge-coupled device*, también conocido como dispositivo de cargas interconectadas). Este dispositivo consiste en un circuito integrado que contiene un arreglo de capacitores con fotodiodos sensibles a la luz. Las imágenes son proyectadas por un lente al arreglo de capacitores. Cada capacitor acumula una carga eléctrica proporcional a la intensidad de luz recibida. El último capacitor del arreglo envía la carga a un amplificador que convierte la energía en voltaje. Por último un circuito de control convierte el contenido del arreglo en un voltaje variante, el cual muestrea, digitaliza y lo almacena en memoria. Estas imágenes almacenadas pueden ser enviadas a una impresora, a un dispositivo de almacenamiento o a algún dispositivo de visualización.

Las aplicaciones de visión por computador toman como entrada los datos enviados por los dispositivos de adquisición de imágenes. Cada aplicación es distinta y tendrá diferentes requerimientos. Es muy importante, al empezar cualquier proyecto de este tipo, analizar el hardware que requerirá la aplicación y las características del mismo.

2.2 Procesamiento digital de imágenes

La meta de toda aplicación de visión por computador es tomar decisiones basándose en los objetos de una escena captada en una imagen. Para poder tomar estas decisiones es importante realzar u obtener información de interés de las imágenes captadas. Esta tarea se la realiza utilizando el procesamiento digital de imágenes.

En el procesamiento digital de imágenes se utilizan un conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información. Las técnicas que se apliquen, dependerán de la información que se desee obtener.

A continuación describiremos como se representa digitalmente a una imagen, los modelos de color utilizados con mayor frecuencia y las técnicas de procesamiento digital de imágenes utilizadas en este proyecto.

2.2.1 Representación digital de una imagen

La forma más común de representar a una imagen digitalmente es como un arreglo bidimensional. Cada elemento de este arreglo contiene información de un pixel de la imagen. Esta información puede ser muy diversa, aunque normalmente suele ser información del color. En la siguiente imagen se muestra un ejemplo de esta representación.

$$I = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} \\ x_{21} & x_{22} & \cdots & x_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} \end{bmatrix}$$

Figura 2.1: Imagen representada como un arreglo bidimensional

Al elemento x_{ij} del arreglo bidimensional se le llama píxel. Cada píxel contiene el valor del tono de gris que se ha asociado a la coordenada i, j al momento de la digitalización. La información de los píxeles está representada por el número de bits utilizados para representar el tono de gris. Mientras mayor sea el número de bits que se utilicen se podrá obtener una mejor representación de la imagen. En la siguiente tabla se muestran los formatos más utilizados.

# de bits	Descripción
1	Imagen monocromática / Blanco y negro
4	Imagen de 16 niveles de gris
8	Imagen de 256 niveles de gris

Tabla 2.1: Formatos del píxel

Para representar el color en una imagen, a cada píxel se le asocian varios componentes. Cada componente tiene asociado un valor que al combinarlos forman el color. El número de componentes, el significado de cada uno de ellos y la manera en la que se los combina para formar el color dependerá del modelo de representación de colores utilizado.

2.2.2 Modelos de representación de colores

Aunque existe gran variedad de modelos de representación de colores, en este trabajo nos vamos a centrar únicamente en dos de ellos: El modelo RGB y el HSV. RGB es un modelo utilizado muy frecuentemente. La mayoría de dispositivos como cámaras de video o cámaras fotográficas utilizan este modelo de representación de colores. Mientras que HSV es un modelo aplicado más en el área de visión por computador. Esto se debe a la manera en que sus

componentes separan la información del color, facilitando ciertas tareas en este tipo de aplicaciones.

2.2.2.1 Modelo RGB

RGB es un modelo que utiliza tres componentes: R (del inglés *red*, “rojo”), G (del inglés *green*, “verde”) y B (del inglés *blue*, “azul”). Los colores se forman realizando una combinación lineal de los componentes del modelo. Es decir que un color es representado por la cantidad de rojo, verde y azul que contenga.

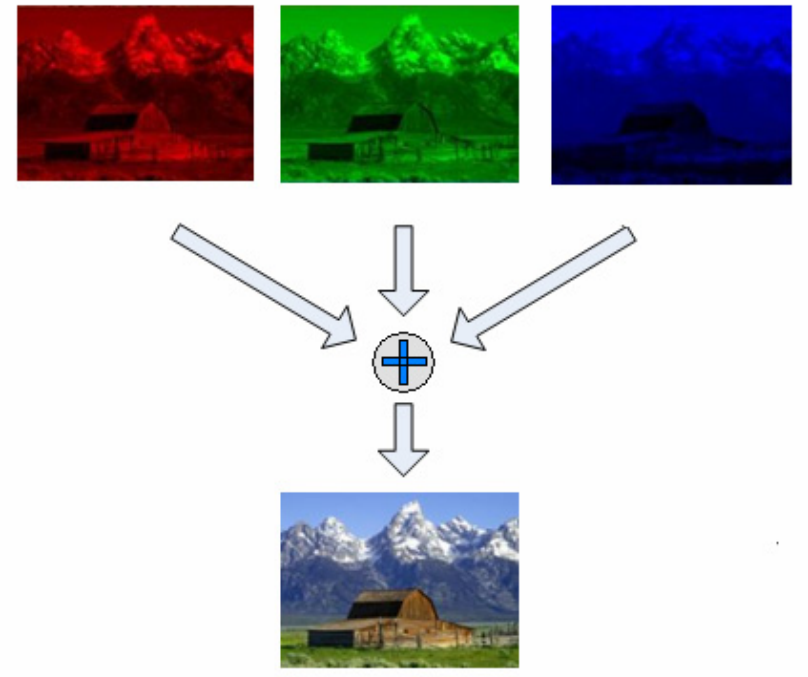

Figura 2.2: Representación de una imagen mediante el modelo RGB

2.2.2.2 Modelo HSV

HSV también es un modelo de tres componentes. El primer componente H viene del inglés *hue* que significa matiz. El matiz indica el tipo color en su forma más básica, se podría decir que representa al color puro.

El segundo componente S viene del inglés *saturation* que significa saturación. La saturación representa la pureza que tiene el color. A medida que un color sea menos saturado será un color más impuro teniendo una apariencia que tiende hacia el gris.

El último componente V viene del inglés *value* que significa valor. El cual contiene información de la cantidad de brillo que tiene el color.

En este modelo un color se define como: El color en su forma más básica, su pureza y su brillo. Combinando estos 3 parámetros se puede representar igual número de colores que en el modelo RGB.

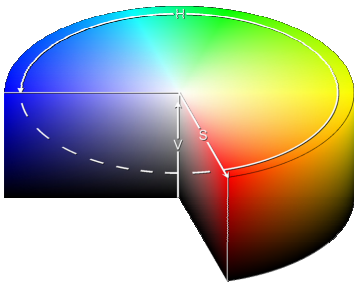

Figura 2.3: Representación de colores mediante el modelo HSV

2.2.2.3 Conversión de RGB a HSV

Como se mencionó anteriormente, en la actualidad, la mayoría de dispositivos utilizan el modelo RGB para representar los colores. También se mencionó que HSV es un modelo que resulta muy útil para el campo de visión por computador. Por estos motivos la conversión o transformación de un modelo a otro resulta una tarea importante a realizar.

Las fórmulas para convertir un color representado en el modelo RGB a uno representado en el modelo HSV son las siguientes:

$$H = \begin{cases} \text{undefined}, & \text{if } MAX = MIN \\ 60 \times \frac{G-B}{MAX-MIN} + 0, & \text{if } MAX = R \\ & \text{and } G \geq B \\ 60 \times \frac{G-B}{MAX-MIN} + 360, & \text{if } MAX = R \\ & \text{and } G < B \\ 60 \times \frac{B-R}{MAX-MIN} + 120, & \text{if } MAX = G \\ 60 \times \frac{R-G}{MAX-MIN} + 240, & \text{if } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{if } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{otherwise} \end{cases}$$

$$V = MAX$$

Donde: MAX es el máximo valor de (R, G, B), MIN es el valor mínimo de (R,G,B), H es la Matiz, S la saturación, V el valor o intensidad, G es Verde, B es Azul, R es Rojo.

2.2.3 Regiones de interés

En procesamiento digital de imágenes se conoce como región de interés (ROI, *Region of interest*) a un área o zona en particular de una imagen que resulta de interés para la realización de algún procesamiento. Esta región suele ser definida con un rectángulo o un conjunto de vértices. A continuación se muestra una región de interés de una imagen representada por un cuadrado.

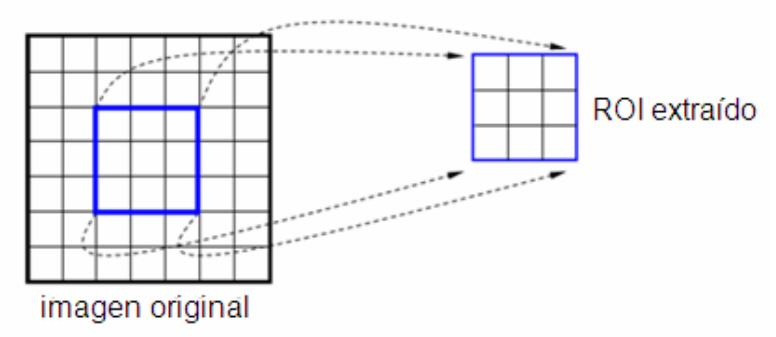

Figura 2.4: Región de interés de una imagen

2.2.4 Binarización de imágenes

La binarización de una imagen es una operación que segmenta la información en dos grupos. Estos grupos se verán identificados en la imagen resultante por dos colores: Blanco y negro. Las zonas con color blanco representarían información que resulta de algún tipo de interés. Mientras que las de color negro serían todo lo contrario.

Para poder segmentar la información es necesario aplicar algún tipo de criterio que indique a cual grupo pertenece. Una de las técnicas más utilizadas es la de definir un umbral (*thresholding*). La idea es que para cada píxel se evalúe si su valor es mayor o igual al del umbral. Si lo es, se asigna el color blanco a la imagen resultante en la misma posición del píxel evaluado. Caso contrario, se asigna el color negro.

El criterio a utilizar dependerá de la aplicación a desarrollar, sin embargo, el resultado de este proceso siempre será una imagen con únicamente dos colores. Es un proceso que reduce la información de manera significativa. Por este motivo suele ser aplicado con la finalidad de utilizarlo en un procesamiento posterior en vez de ser el resultado final en sí.

2.2.5 Operación AND entre imágenes

AND es una operación lógica de punto a punto que se aplica a dos imágenes binarias. El resultado de esta operación es una tercera imagen que contendrá la multiplicación de los valores de los píxeles de las dos imágenes de entrada. La operación se realiza píxel por píxel manteniendo las coordenadas resultantes. Es decir que el valor del píxel en la imagen resultante en la coordenada (i, j) será igual al valor del píxel en la coordenada (i, j) de la primera imagen por el valor del píxel en la coordenada (i, j) de la segunda.

2.2.6 Histogramas de intensidad

Un histograma de intensidad es una gráfica que muestra el número de píxeles que corresponden a un valor de tono en un canal

específico de una imagen. Por ejemplo para una imagen en tono de gris que utiliza 8 bits para representar los colores, el eje X del histograma tendría valores de [0, 255] mientras que el eje Y contendría por cada valor de X la ocurrencia de ese valor de tono de gris en la imagen. Para el caso de imágenes a colores se necesitaría calcular un histograma de intensidad por cada canal.

Los histogramas de intensidad proporcionan información muy diversa acerca de la imagen. Por ejemplo nos podría decir si tiene un buen contraste o no, los colores con mayor ocurrencia o la intensidad general de la imagen. Estas características y otras más se ven evidenciadas en los valores y la distribución de los histogramas de intensidad de una imagen.

2.2.7 Retroproyección

Retroproyección (*backprojection* en inglés) es un proceso que toma como entrada una imagen y un histograma de intensidad y que genera como resultado una nueva imagen. Los valores de los píxeles (i,j) de la imagen resultante son los valores de las frecuencias del histograma de intensidad de los píxeles (i,j) de la imagen original. Es decir que este proceso reemplaza los valores de los píxeles de la

imagen original por los valores de la frecuencia en el histograma de intensidad.

2.3 Algoritmos de reconocimiento y seguimiento de objetos

2.3.1 Algoritmo de desplazamiento medio

El algoritmo de desplazamiento medio (mean-shift) es utilizado para encontrar los picos o máximos locales en un grupo de datos obtenidos del histograma de una muestra. El algoritmo básicamente utiliza un vector para realizar el análisis. En cada iteración se irá moviendo hasta encontrar el punto deseado o hasta llegar a un límite de iteraciones establecido.

Para poder realizar este procedimiento es necesario aplicar al histograma un *kernel* de suavizado. El motivo de esto es que para que el vector se pueda mover libremente la función necesita ser continua y no discreta como es la naturaleza de los histogramas. La elección del *kernel* a aplicar dependerá de la implementación del algoritmo.

2.3.2 Algoritmo de desplazamiento medio continuamente adaptativo

El algoritmo de desplazamiento medio continuamente adaptativo (cam-shift) es una modificación al algoritmo de desplazamiento medio (mean-shift). La diferencia radica en que el algoritmo de desplazamiento medio utiliza una función de probabilidad estática mientras que el adaptativo hace uso de una función de probabilidad variable. Este algoritmo se puede resumir en 5 pasos [5]:

1. Ajustar la región de interés a toda la imagen.
2. Seleccionar la región a ser rastreada.
3. Calcular la distribución de probabilidad del color.
4. Iterar el algoritmo *de desplazamiento medio* para encontrar la centroide de la imagen de probabilidad.
5. Para las siguientes imágenes centrar la ventana de búsqueda a la posición encontrada en el paso 4. Ir al paso 3

2.4 Transmisión de datos de la aplicación al robot

2.4.1 El puerto serie y el estándar RS-232

El estándar RS232 es un protocolo de comunicación serial asincrónico propuesto por la Asociación de Industrias Electrónicas (EIA). El protocolo es asincrónico porque no tiene una señal separada de reloj para sincronizar el envío de datos. En lugar de esto se sincroniza por medio de sus propios datos recibidos, a través de los pulsos de inicio y fin.

Las señales con las que actúa el puerto son digitales (0 - 1) y la tensión a la que trabaja es de +-12 V. En donde tenemos que +12V = Lógica “0” y -12 V = Lógica “1”.

Existen varios tipos de tramas de datos en una comunicación mediante el protocolo RS232. El más común es el que tiene ocho bits de datos, uno de inicio y uno de fin, es decir diez bits en total. Adicionalmente se puede incluir un bit de paridad para la comprobación de errores, en ese caso sería una trama de once bits.

La velocidad de transmisión se mide en baudios (bits/segundo). Entre las velocidades más comunes tenemos 9600, 4800, 2400 y 1200 baudios. Por ejemplo a 9600 baudios, se transmiten 960 bytes cada segundo, considerando que la trama es de 10 bits.

Una de las ventajas de la comunicación serial es que para enviar y recibir datos es posible hacerlo solo con tres cables, uno de transmisión, otro de recepción, y la referencia tierra. Sin embargo el estándar RS232 incluye otros pines con funciones específicas para cada uno. Es por esto que los conectores comerciales son de nueve o de veinticinco pines.

Las características de los pines y sus nombres típicos son:

- TXD: Transmitir datos señal de salida.
- RXD: Recibir datos señal de entrada.
- RTS: Solicitud de envío señal de salida.
- DTR: Terminal de datos listo señal de salida.
- CTS: Libre para envío señal de entrada.
- DSR: Equipo de datos listo señal de entrada.
- DCD: Detección de portadora señal de entrada.
- SG: Tierra referencia para señales.
- RI: Indicador de llamada señal de entrada.

Antes de iniciar cualquier comunicación con el puerto RS232 se debe de determinar el protocolo a seguir dado que el estándar del protocolo no permite indicar en qué modo se está trabajando, es la

persona que utiliza el protocolo la que debe decidir y configurar ambas partes antes de iniciar la transmisión de datos [6]. Siendo los parámetros a configurar los siguientes:

- Protocolo serie (bits de datos, paridad, bits fin).
- Velocidad del puerto (baudios).
- Protocolo de control de flujo (RTS/CTS o XON/XOFF).

2.4.2 Transmisión de datos por radiofrecuencia

Por lo general las formas más comunes de enviar datos en aplicaciones de robótica son mediante infrarrojos o radiofrecuencia. Para el caso de robots móviles resulta más conveniente el uso de dispositivos de radiofrecuencia, ya que ellos, a diferencia de los dispositivos infrarrojos, no necesitan tener línea de vista entre los dispositivos que se comunican. Sin embargo cuando se pierde la línea de vista entre el emisor y el receptor, la distancia máxima que estos pueden estar separados disminuye [7].

Es posible diseñar e implementar por completo un sistema de comunicación por radiofrecuencia. Así como también es posible utilizar módulos RF comerciales. Para el control de robots móviles la

opción de usar módulos comerciales resulta más conveniente, tanto en tiempo de desarrollo como en costo.

A continuación se detallan las características principales de dos módulos comerciales de radiofrecuencia, el TRLP434 y el SRF418L.

2.4.2.1 Transmisor/Receptor TLP/RLP 434

El TLP434 y el RLP434 son módulos de radiofrecuencia, transmisor y receptor respectivamente, que operan a 433.92MHz. Estos dispositivos ofrecen una comunicación unidireccional, del transmisor al receptor, por ejemplo del punto A al B. En caso de que una aplicación requiera una comunicación bidireccional, digamos del punto A al B y también de B a A, será necesario usar otro par de módulos de distinta frecuencia para la transmisión de B a A.

Una de las ventajas de estos módulos es que para incluirlos en una aplicación, simplemente se requiere que haya una interfaz serial. El módulo transmisor tiene un pin de entrada de datos, por el cual ingresa de manera serial la información que será transmitida. En el módulo receptor así mismo hay un pin de salida serial, en el cual se obtienen los datos que llegan desde el emisor.

La velocidad máxima de transmisión es de 2400bps. Hay que tener en cuenta esto a la hora de configurar el protocolo serial, ya que si los datos son enviados a más de 2400 baudios, no llegarán correctamente al receptor. Por lo tanto es preferible trabajar a una velocidad menor, la cual podría ser de 1200 baudios.

La distancia máxima de separación entre el transmisor y el receptor es de aproximadamente cien metros. Es recomendable que el camino entre ambos esté despejado, ya que mientras mayores son los obstáculos, mayor será la atenuación de la señal. Lo que disminuirá la distancia máxima que podrán estar separados.

El TLP434 y el RLP434 usan modulación ASK. Ese tipo de modulación digital es análoga a AM en comunicaciones analógicas, en donde se codifica la información variando la amplitud de la señal portadora. Por lo tanto, la señal de radio es conmutada entre nivel alto y bajo de la potencia de salida para codificar el flujo de los pulsos de datos .

La transmisión mediante modulación ASK es muy susceptible a interferencias [8]. El receptor interpreta la intensidad de la señal como información, por lo que el ruido eléctrico que es sumado a la

señal transmitida, produce errores en la recepción de los comandos. Sin embargo este tipo de modulación tiene sus ventajas como su simplicidad y bajo costo.

En cuanto a la alimentación, los módulos funcionan con 5V. Sin embargo tienen un margen de tolerancia de 0.5V. Esto lo hace fácil de ser incluido en aplicaciones electrónicas que usen niveles de voltaje TTL.

2.4.2.2 Transmisor/Receptor SRF-418L

Este módulo de comunicación por radiofrecuencia contiene internamente un transmisor y receptor, es decir, es útil para aplicaciones donde sea necesaria una comunicación bidireccional. El control para que el dispositivo funcione como transmisor o como receptor se lo realiza mediante el nivel de voltaje en un pin de entrada específico.

La frecuencia a la que opera el dispositivo va desde los 414.6MHz hasta los 426.6MHz. Existen trece canales distintos de comunicación entre las frecuencias mencionadas. La ventaja de esto es que se puede cambiar fácilmente la frecuencia a la que operan los

dispositivos, en caso de que se presenten problemas de interferencia en algún canal [9].

La modulación digital utilizada por estos dispositivos es FSK. La cual es análoga a FM para comunicaciones analógicas. Este tipo de modulación usa la variación de la frecuencia de la señal portadora para codificar los datos que se envían. Esto hace que el ruido no afecte tanto al receptor, ya que la variación en la amplitud no representa nada, contrario a lo que sucede en la modulación ASK [10]. Adicionalmente, estos dispositivos soportan mayores velocidades de transmisión, la cual es hasta de 19200bps.

Los módulos RF trabajan con 5V, pero al utilizar modulación FSK consumen más corriente que los que usan ASK. Otra desventaja es su mayor costo. Sin embargo si una aplicación requiere comunicación bidireccional o menor susceptibilidad al ruido, es recomendable el uso de este tipo de dispositivos.

2.5 Robot Móvil

En esta sección se enumeran los componentes del robot móvil. El cual consta de motores, microcontrolador, batería y dispositivos

periféricos. A continuación se mencionan las características principales de cada uno de ellos.

2.5.1 Motores del robot

En las aplicaciones con robots móviles, los motores son un componente fundamental. Estos son los que hacen que el robot se pueda movilizar. Ya sea que se utilicen ruedas, patas o algún otro dispositivo, es necesario el uso de motores para poder mover dichos mecanismos.

Existen varios tipos de motores, los de corriente directa, los de paso a paso y los servomotores. Estos últimos son simplemente un motor de corriente directa con un lazo cerrado de control para asegurar su posicionamiento exacto. Además existen los motores de reluctancia variable, los motores DC sin escobillas, síncronos, de corriente alterna, etc.

Los motores comúnmente usados en aplicaciones de robots móviles son los de corriente directa y los de paso a paso. A continuación se presentan las características de cada uno de ellos.

2.5.1.1 Motor de corriente directa

El motor de corriente directa (DC) es uno de los más versátiles en la industria. Su fácil control de posición, par y velocidad lo han convertido en una de las mejores opciones en aplicaciones de control y automatización de procesos.

Accionar un motor DC es muy simple y solo es necesario aplicar la tensión de alimentación entre sus bornes. Para invertir el sentido de giro basta con invertir la alimentación y el motor comenzará a girar en sentido opuesto.

A diferencia de los motores de pasos y los servomecanismos, los motores DC no pueden ser posicionados y/o enclavados en una posición específica. Estos simplemente giran a la máxima velocidad en caso de recibir la tensión nominal.

Una buena manera de controlar la velocidad de un motor DC es mediante un modulador de ancho de pulso. Si la aplicación requiere un controlador a lazo cerrado se puede incluir un codificador para que realimente al sistema la velocidad a la que gira el motor.

2.5.1.2 Motor de pasos

Los motores de pasos son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos. La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8°. Es decir, que se necesitarán 4 pasos en el primer caso (90°) y 200 para el segundo caso (1.8°), para completar un giro completo de 360°.

Estos motores poseen la habilidad de poder quedar enclavados en una posición o bien totalmente libres. Si una o más de sus bobinas están energizadas, el motor estará enclavado en la posición correspondiente. Por el contrario quedará completamente libre si no circula corriente por ninguna de sus bobinas.

Básicamente estos motores están constituidos normalmente por un rotor sobre el que van aplicados distintos imanes permanentes y por un cierto número de bobinas excitadoras en su estator. Toda la conmutación, o excitación de las bobinas, deber ser externamente manejada por un controlador.

Los motores de pasos, tienen un alto torque de detención. Así como también tienen un alto torque cuando giran a bajas revoluciones. Mientras aumenta la velocidad del rotor, disminuye el torque. Lo contrario sucede con los motores de corriente directa, de los cuales se obtiene bajo torque cuando giran a bajas velocidades y el torque va aumentando mientras aumenta la velocidad [11].

Existen dos tipos de motores de pasos de imán permanente. Ellos son el bipolar y el unipolar.

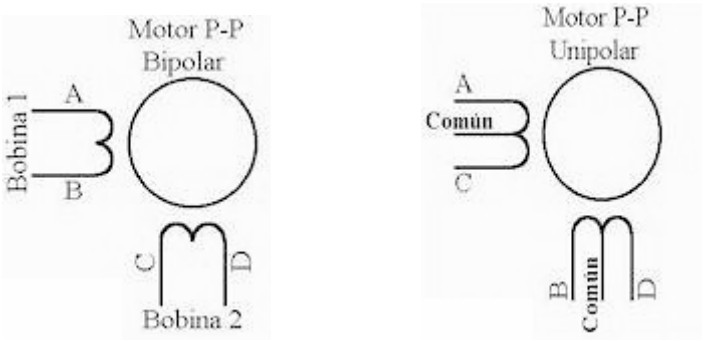

Figura 2.5: Motor de pasos bipolar y unipolar

Los motores bipolares tienen generalmente cuatro cables de salida. Requieren del cambio de dirección del flujo de corriente a través de las bobinas en la secuencia apropiada para realizar un movimiento.

Los motores unipolares suelen tener seis o cinco cables de salida, dependiendo de su conexión interna, sin embargo tienen cuatro bobinas en total. Este tipo de motores se caracteriza por ser más simple de controlar. Para moverse paso por paso, requiere que sus bobinas sean energizadas en una secuencia específica [12].

Una de las secuencias de encendido de bobinas en un motor de pasos se muestra en la siguiente tabla. Esta secuencia tiene la ventaja de tener un alto torque de paso y de retención:

PASO	Bobina A	Bobina B	Bobina C	Bobina D
1	Encendido	Encendido	Apagado	Apagado
2	Apagado	Encendido	Encendido	Apagado
3	Apagado	Apagado	Encendido	Encendido
4	Encendido	Apagado	Apagado	Encendido

Tabla 2.2: Secuencia de encendido de bobinas (alto torque)

Otra de las secuencias que se puede usar para mover un motor de pasos se muestra en la siguiente tabla. Al estar encendida solo una bobina a la vez, ésta secuencia consume menos corriente, aunque también produce un menor torque de paso y de detención.

PASO	Bobina A	Bobina B	Bobina C	Bobina D
1	Encendido	Apagado	Apagado	Apagado
2	Apagado	Encendido	Apagado	Apagado
3	Apagado	Apagado	Encendido	Apagado
4	Apagado	Apagado	Apagado	Encendido

Tabla 2.3: Secuencia de encendido (bajo consumo de corriente)

2.5.2 Control del robot mediante microcontroladores

El elemento principal que gobierna las acciones del robot es el microcontrolador. Este dispositivo posee grabadas internamente unas subrutinas que ejecutarán las acciones que el usuario haya programado.

El microcontrolador es un dispositivo que internamente posee CPU (Unidad de procesamiento central), memoria de programa, memoria de datos y periféricos como temporizadores, controlador de puerto serie, etc. Es decir, este dispositivo es un controlador embebido en

un solo integrado. Esto hace que sea ampliamente utilizado para aplicaciones de robótica. Además, como ventajas se pueden mencionar su reducido tamaño, bajo costo y facilidad para incluirlo en prototipos electrónicos.

2.5.2.1 PIC 16F877A

Este dispositivo es un microcontrolador que posee tecnología CMOS, es de baja potencia y alta velocidad resistente a la estática y a temperaturas industriales su voltaje de operación está entre 2.0V y 5.5V. Se lo puede programar en ensamblador o lenguajes de alto nivel como C o Basic. Tiene 368x8 bytes de memoria RAM 256x8 bytes de memoria EEPROM. Entre sus principales características tenemos:

Temporizadores:

- Timer0: usa 8 bits y puede ser usado como temporizador o contador.
- Timer1: usa 16 bits, puede ser usado como temporizador o contador y puede ser incrementado en modo de hibernación con una señal de reloj externa.

- Timer2: usa 8 bits y puede ser usado como temporizador o contador con 8 bits de registro de periodo.

Dos módulos que pueden ser usados como:

- Módulo de captura de máximo 16bits, cuya resolución es 12.5ns.
- Módulo de comparación de máximo 16bits, cuya resolución es 200ns.
- Módulo de modulación de ancho de pulso (PWM) cuya resolución es 10bits.

Comunicación:

- Sincrónica con el puerto serie (SSP) con SPI, I2C.
- Transmisor/Receptor universal sincrónico y asincrónico (USART/SCI) con 9 bits de detección de dirección.

Convertidor analógico-digital:

- Resolución de 10 bits.
- Hasta 8 canales de entradas analógicas.

2.5.2.2 ATMEGA8535

Este dispositivo es un microcontrolador resistente a la estática cuyo voltaje de operación está entre 4.5V y 5.5V. Tiene 512 bytes de

memoria EEPROM y 512 bytes internos de SRAM. Entre sus principales características tenemos:

Temporizadores:

- Dos temporizadores de 8 bits que pueden ser usados como contadores.
- Un contador de 16 bits.
- Un contador de tiempo real separado del oscilador.

Módulos:

- Cuatro módulos de modulación de ancho de pulso (PWM) cuya resolución es de 10bits.

Comunicación:

- Sincrónica con el puerto serie (SSP) con SPI, I2C.
- Transmisor/Receptor universal sincrónico y asincrónico (USART/SCI) con 9 bits de detección de dirección.

2.5.3 Dispositivos periféricos del robot

Como periféricos tenemos los sensores y dispositivos secundarios que aumentan las funcionalidades del robot. Por lo general los sensores ayudan al robot a interaccionar con el medio. Estos son, por ejemplo, los sensores de contacto, de proximidad, de

temperatura, leds, zumbador, etc. A continuación se mencionan unos dispositivos periféricos que podrían ser incluidos en distintas aplicaciones de robots móviles.

2.5.3.1 Sensores de contacto

Los sensores de contacto nos indican simplemente si ha habido contacto o no con algún objeto, sin considerar la magnitud de la fuerza de contacto [13]. Al existir contacto, este cierra un circuito eléctrico y permite el flujo de corriente a través del sensor. El dispositivo detecta este flujo de corriente y así se entera de que el botón ha sido presionado. Al soltar el botón, el circuito se abre y cesa el flujo de corriente [14].

Estos sensores suelen ser interruptores de límite o micro interruptores, que son sencillos dispositivos eléctricos que cuando se contacta con ellos cambian de estado.

2.5.3.2 Zumbador

Un zumbador es una bocina que emite un sonido continuo, el cual puede ser controlado por un circuito integrado. El zumbador no

posee en su estructura electrónica el circuito que hace que un timbre emita siempre el mismo sonido y es por esto que necesita la entrada del circuito integrado u otra fuente; esto lo hace capaz de emitir diferentes sonidos y tener un gasto de corriente muy bajo puede ser usado con baterías de 1.5V también hay de 125V alterna 60Hz [15].

2.5.4 Batería y regulador de voltaje del robot

Las baterías generan electricidad a partir de reacciones químicas. En una batería hay un compartimiento con exceso de electrones (el polo positivo) y otro al que le faltan (el polo negativo). Al conectar un dispositivo, la corriente eléctrica fluye de un polo a otro hasta que la diferencia de electrones se equilibra. En ese momento, la batería está descargada.

La ventaja de las baterías recargables es que haciendo pasar de nuevo una corriente eléctrica por ellas, se restablece la diferencia de cargas y la batería se puede usar de nuevo. Las baterías recargables más antiguas son las de plomo y ácido, que se siguen utilizando en los automóviles. En dispositivos electrónicos, las más usadas son las de hidruro de níquel-metal (Ni-MH) y las de iones de litio (Li-Ión)

2.5.4.1 Batería de Litio-lón (Li-Ion)

Las baterías litio-ión pueden almacenar una gran densidad de energía en poco espacio. Esto es posible debido a que el litio es el más liviano de todos los metales, posee el mayor potencial electroquímico y representa el mayor contenedor de energía. La vida típica de una batería de LI-Ion es de 300 a 500 ciclos de carga/descarga o de dos años desde su fabricación.

Estas baterías nos proporcionan algunas ventajas con respecto a sus antecesoras de níquel-metal (Ni-MH) o Níquel-Cadmio (Ni-Cd). Las baterías de litio-ión no sufren el efecto memoria (no necesitan ser descargadas por completo antes de volver a cargarlas para mejorar su ciclo de vida útil). Además, necesitan poco mantenimiento y su auto descarga es menor a la mitad de la que sufren las baterías de Ni-Cd y Ni-MH.

Como desventaja en las baterías de Litio-Ion podríamos decir que estas envejecen aunque no se las esté usando incluso cuando está en su empaque. Para reducir esto las baterías deben estar en un lugar fresco y a un 40% de su carga [16].

2.5.4.2 Batería de Hidruro de Níquel-metal (Ni-MH)

Las baterías de Ni-MH son más respetuosas con el medio ambiente que su antecesora la batería de Ni-Cd. Además pueden almacenar un 30% más de energía que esta, por lo tanto la carga dura más tiempo.

Su desventaja principal es que sufren del efecto memoria, necesitan descargarse por completo al menos una vez al mes. Esto se debe a que los elementos activos de estas se encuentran en forma de cristales. Cuando estas baterías se cargan sin descargarlas del todo, se forman grupos de cristales que reducen la capacidad. Por este motivo hay que realizar una primera carga larga (de unas 15 horas) con la batería totalmente descargada. Después conviene descargar totalmente la batería al menos una vez por semana [17].

2.5.4.3 Regulador de voltaje LM2575

El LM2575-5 nos da 5V a la salida y representa una mejor opción frente al popular regulador lineal de tres terminales. Debido a su alta eficiencia reduce significativamente el tamaño del disipador y en algunos casos este no es requerido. Este regulador ha sido creado

para usarse con inductores disponibles de diferentes fabricantes. Una gran ventaja de este regulador es que facilita el diseño de fuentes de poder, usando de cuatro a seis componentes externos para su funcionamiento.

2.5.4.4 Regulador de voltaje LM7805

La serie LM78XX nos permite voltajes de 5, 6, 8, 9, 10, 12, 15, 18, 24V, en este caso el LM7805 nos da a la salida de 4.8V a 5.2V. Lo más importante de este dispositivo es que con la configuración adecuada podemos controlar la corriente de salida para proteger a los equipos que este alimente, y con un disipador adecuado los LM78XX pueden entregar hasta un amperio a la salida.

La desventaja de este tipo de reguladores lineal es que tiene una menor eficiencia comparada con el LM2575. Sin embargo es ampliamente utilizado en aplicaciones donde no se requiere mucha eficiencia en la energía consumida.

CAPÍTULO 3

3 DISEÑO DE LA SOLUCIÓN

En el presente capítulo se describe el diseño del proyecto a implementar. En principio detallaremos el diseño general de la solución en donde se especifica cómo interaccionan sus diferentes componentes. Por último se describirán las especificaciones funcionales tanto de la aplicación como del hardware a implementar.

3.1 Diseño general

El presente proyecto tiene como finalidad la construcción de un robot móvil que será teleoperado por una aplicación. Esta aplicación utilizará técnicas de procesamiento digital de imágenes y visión por computador para reconocer y seguir objetos tangibles, los cuales serán manipulados por el usuario. Los objetos físicos que el usuario manipulará serán figuras geométricas de distintos colores, cada una de ellas representará un comando de control para el robot. La

aplicación detectará el movimiento de estos objetos utilizando una cámara de video. Luego, enviará la orden al robot de acuerdo al movimiento y objeto detectado.

Para poder realizar lo anterior, se diseñará un algoritmo de reconocimiento y seguimiento de objetos. Este algoritmo será de buena precisión, lo cual reducirá el ruido comúnmente presente en este tipo de aplicaciones, disminuyendo el envío de comandos erróneos basados en información incorrecta al robot.

Para el control del robot se desarrollarán e implementarán algoritmos en un microcontrolador. El cual se encargará de recibir los comandos enviados por la aplicación y gobernar los motores y dispositivos periféricos del robot.

A continuación se presenta un diagrama de bloques del proyecto.

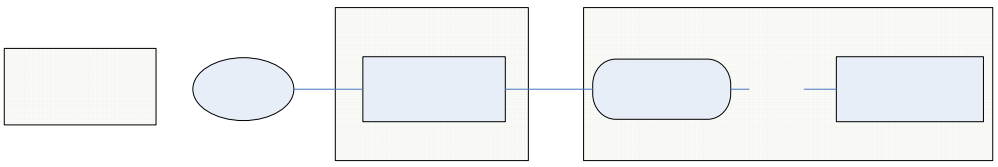

Figura 3.1: Diagrama de bloques del proyecto

3.2 Diseño de la aplicación

A continuación se detalla el diseño de la aplicación a desarrollar. Se describen las funcionalidades que debe ofrecer el software. Las características y requisitos que necesita tener el algoritmo de reconocimiento y seguimiento de objetos. También se especifica el protocolo de comunicación entre la aplicación y el robot.

3.2.1 Especificaciones funcionales

La funcionalidad principal de la aplicación a desarrollar será la de interpretar los comandos que el usuario realice mediante el movimiento de objetos físicos de distintos colores. Para esto se deberá tener información de los objetos a considerar. Es decir, la aplicación debe conocer:

- Que objetos buscar.
- Como reconocer dichos objetos (características del color).
- Que representa cada uno de ellos.

Por ejemplo, el objeto de color verde representaría la velocidad del motor izquierdo, mientras que el objeto de color rojo, la velocidad del

motor derecho. El movimiento de ellos por parte del usuario deberá ser interpretado por la aplicación como un aumento o disminución de velocidad del motor izquierdo o derecho.

También es importante dar la libertad de poder alimentar al sistema con la información del color de los objetos a considerar. Se debería dar la facilidad de configurar por cada comando que la aplicación pueda enviar el robot los datos del objeto que representa dicho comando. El motivo de esto es que la información del color que recibe la aplicación puede variar por una serie de factores. Los más comunes son dos:

1. Las características de la cámara de video: Cada cámara de video tiene diferentes características como calidad de imagen, cuadros por segundo, etc. De esto dependerá la calidad del color que la cámara capte de la escena. A medida que la cámara sea de mayor calidad, mayor será la similitud del color del objeto que vemos en la imagen comparado con el color del objeto que vemos en la escena real. Por ejemplo si captamos una misma escena con diferentes cámaras de video una podría generar imágenes con colores más claros mientras que la otra con colores opacos.

2. La iluminación de la escena: Los colores de los objetos en la escena que captará la cámara de video dependerán de la iluminación. A medida que la iluminación disminuya los colores se verán degradados viéndose cada vez más opacos.

Por último, la aplicación deber dar una retroalimentación al usuario de lo que está haciendo. Más allá de que si el usuario manipula al objeto y ve inmediatamente la acción que debería efectuar el robot, es importante que en la interfaz de la aplicación también se vea la interpretación que ha hecho el algoritmo de la acción realizada por el usuario. Esto será de gran utilidad al momento de que se estén configurando los colores de los objetos que representan los comandos del robot.

3.2.2 Algoritmo de reconocimiento y seguimiento de objetos

El algoritmo de reconocimiento y seguimiento de objetos deberá ser implementado pensando en que esta es una aplicación en tiempo real. El usuario necesita ver una respuesta casi inmediata en las acciones del robot cuando manipule algunos de los objetos de interés. Es decir, necesita ser un algoritmo lo suficientemente rápido.

Otra característica importante es que sea preciso, en vista de que las acciones que el robot realice dependerán del resultado de este algoritmo. Para lograr este objetivo, se necesita hacer verdaderos esfuerzos en la reducción del ruido presente en toda señal de video y en el análisis de los datos. De esta forma se podrá reconocer los objetos de interés. Como este algoritmo también deberá realizar seguimiento a los objetos, la detección inicial debe ser libre de fallos para que estos no sean propagados en las siguientes iteraciones.

Estas dos características que son requeridas, podrían estar en conflicto. Implementar un algoritmo más preciso involucraría realizar más cálculos en general. Cálculos que podrían consumir muchos recursos del computador y que afectarían a su rapidez de manera significativa. Es decir un algoritmo sumamente preciso podría tener graves problemas de velocidad. Por esta razón es necesario encontrar un balance que se ajuste a las necesidades particulares de este proyecto.

3.2.3 Protocolo de transmisión de datos al robot

Es necesario enviar los comandos de control desde el computador al robot de manera inalámbrica. Para esto se debe estructurar una

trama de datos que se encargará de llevar toda la información necesaria para el correcto funcionamiento del robot.

Los datos incluidos en esta trama son: La velocidad del motor izquierdo, la velocidad del motor derecho y las órdenes de las acciones que puede realizar el robot. Estos datos están conformados por bytes. En nuestra aplicación tenemos tres bytes principales, dos para las velocidades de los motores y uno para las órdenes de las acciones.

Adicionalmente la trama de datos debe incluir un byte de inicio. El mismo que sirve para que el receptor sepa que a partir de ese byte, los siguientes que se le envían serán los de información.

Por lo general ocurren errores en la transmisión, es decir el byte recibido no es igual al enviado. Esto lleva a buscar alguna forma de minimizar este riesgo de trabajar con datos erróneos. En nuestro caso, cada uno de los tres bytes principales de información se lo envía dos veces consecutivas. Si el receptor encuentra que ambos bytes son iguales, quiere decir que ese dato es válido, en el caso de que sean distintos, el dato no es válido, por lo que no es tomado en

cuenta y se espera el siguiente envío de información para repetir el proceso.

La trama de datos enviada consta en total de siete bytes, uno de inicio y seis de datos. Esta trama se la envía continuamente, lo que facilita la rápida actualización de las órdenes que se le envían al robot.

En resumen, el protocolo de comunicación diseñado empieza con el envío del byte de inicio. Luego, el byte de la velocidad del motor izquierdo es enviado dos veces, al igual que el byte de la velocidad del motor derecho y el de las órdenes de las acciones. El controlador del receptor una vez que ha ingresado el byte de inicio, almacena los seis bytes siguientes que le llegan y los compara de dos en dos. De esta forma el controlador verifica que los datos de cada par de bytes sean iguales, lo que disminuye las probabilidades de trabajar con datos erróneos.

La siguiente figura muestra la estructura de la trama de datos enviada por la aplicación al robot.

Figura 3.2: Estructura de la trama de datos enviada

3.3 Diseño de los componentes de hardware

Byte Inicio Byte Motor Izquierdo Byte Mo Izquierc

A continuación se detalla el diseño del hardware a implementar. Se describen las funcionalidades que deberían tener tanto el robot como el circuito transmisor de los comandos. Además se especifican sus características y los requisitos que deben cumplir.

3.3.1 Especificaciones funcionales

El tipo de robot a desarrollarse es uno móvil, de ruedas. Deber tener facilidad para poder desplazarse tanto hacia adelante como hacia atrás y ser capaz de realizar giros. Además se requiere que tenga al menos cuatro velocidades en cada dirección de giro.

Es necesario que el computador que corre la aplicación y el robot no se comuniquen por medio de cables. El envío de órdenes de la aplicación al robot debe ser inalámbrico. De esta forma se le da

mayor libertad al robot de moverse fácilmente y se evita cualquier obstrucción que el cable pueda causar.

El robot además tendrá como dispositivo periférico un zumbador. El cual va a ser accionado por el usuario por medio de la aplicación. Este zumbador actuará solamente cuando el usuario lo desee, es decir no depende del estado en que se encuentre el robot.

Otras de las opciones que tiene el robot es la de detener los motores. Esta opción hará detener la marcha del robot sin importar la velocidad o dirección en la que estaba moviéndose. Esta opción es controlada por la aplicación según el usuario lo ordene.

Adicionalmente el robot tendrá un led indicador. Este led se encenderá cuando ambos motores del robot estén detenidos. Específicamente cuando el usuario ordene que ambos motores tengan velocidad cero.

3.3.2 Robot móvil

Para el diseño de la parte de hardware del proyecto se consideró que se necesita hacer un robot móvil de ruedas. Para esto se tiene en

cuenta que aunque hay varias configuraciones de robots móviles, la más versátil es la de control diferencial, la misma que usa dos ruedas principales laterales. Esta configuración permite mayor facilidad para realizar los giros, ya que incluso puede rotar sobre su propio eje [18].

Para poder controlar estas dos ruedas principales es necesario usar un motor por rueda. Es decir van a ser necesario usar dos motores independientes. Gracias a esto el robot podrá moverse hábilmente en todas las direcciones.

Aunque el control del robot se lo pudo haber realizado desde el computador, se prefirió usar un microcontrolador para esta tarea. Se decidió el uso de un microcontrolador debido a que este se encuentra dentro del robot y el tiempo que se tomará decidir lo que debe hacer será menor. Además en caso de que se pierda la comunicación por un instante no se afectará la ejecución de la tarea ordenada al robot. En este caso el robot continuará realizando la acción que estaba haciendo anteriormente.

Debido a que el robot requiere movilizarse sin estar conectado a nada fijo, la fuente de poder necesita ser portátil y ser llevada dentro del robot. Para esto es necesario usar una batería para energizar al

robot. Esta batería se encargará de alimentar tanto a los componentes electrónicos, como a los motores del robot.

Además, es necesario incluir un módulo de recepción inalámbrico dentro del robot. Este módulo se encargará de recibir las señales que envía el transmisor de la aplicación y hacérselas llegar al microcontrolador.

CAPÍTULO 4

4 IMPLEMENTACIÓN

Este capítulo se dedica a detallar como se implementó la solución propuesta. A continuación se mencionan las herramientas seleccionadas tanto para la implementación del software como del hardware. Además en este capítulo se menciona el proceso de implementación de la aplicación y del robot.

4.1 Herramientas seleccionadas para la implementación

4.1.1 Software

El software implementado es una aplicación de escritorio para la plataforma Windows. El cual tiene los siguientes requisitos:

- Tener instalado el sistema operativo Windows, mínimo la versión Millenium.

- Contar con un mínimo de 128MB de memoria RAM, pero se recomienda 256MB.
- Tener un puerto usb para poder conectar la cámara web.
- Tener un puerto serial para conectar el transmisor de datos.
- Ratón y teclado para manejar la aplicación.
- Monitor para poder visualizar la retroalimentación que ofrece la aplicación.

Las herramientas que seleccionamos para el desarrollo de la aplicación son las siguientes:

- Lenguaje de programación C++
- Librería OpenCV
- Winapi

Seleccionamos C++ por ser uno de los lenguajes de programación más robustos que existen en la actualidad. Abarca los tres paradigmas de programación que utilizaremos en este proyecto: la programación estructurada, programación genérica y programación orientada a objetos. Además de ser un lenguaje rápido en comparación a otros que existen en el mercado como por ejemplo Java o C#.

OpenCV es una librería desarrollada por Intel que contiene una seria de funciones y estructuras que son utilizadas para aplicar las operaciones más comunes de procesamiento digital de imágenes y visión por computador. Tiene una documentación muy extensa y hay mucho soporte en internet por medio de grupos de discusión. Seleccionamos esta herramienta por la funcionalidad que ofrece y porque ha sido utilizada y probada en un gran número de proyectos.

Por último, para el desarrollo de interfaces gráficas en C++ comúnmente se utiliza una de estas 2 librerías: Winapi o MFC. Ambas ofrecen las herramientas necesarias para crear ventanas, cuadros diálogos, controles etc. La diferencia radica en que Winapi utiliza un paradigma genérico estructurado mientras que MFC es orientado a objetos. Seleccionamos Winapi por preferencia más no a ningún otro tipo de ventaja que ofrezca el uno sobre el otro.

4.1.2 Hardware

A continuación se mencionan los componentes y dispositivos seleccionados para la implementación del hardware, tanto para el robot móvil como para el circuito transmisor de datos de la aplicación

al robot. Además, se mencionan las razones principales por las que se escogieron dichos componentes para la implementación.

4.1.2.1 Robot Móvil

El robot móvil implementado usa el PIC 16F877A como controlador principal. Debido a que este microcontrolador es una computadora embebida en un solo chip, para su funcionamiento no requiere de componentes adicionales de hardware como memorias o controladores de periféricos [19]. Esto hace que sea fácil incluirlos en circuitos de aplicaciones electrónicas o hacer prototipos donde ellos sean el controlador principal.

Para el caso de nuestro robot en particular, el PIC 16F877A, al poder trabajar con una señal de reloj de 20MHz [20], garantiza rapidez en la ejecución de los algoritmos de control. Así como el número de entradas y salidas que tiene el PIC es suficiente para las que se necesitan en la implementación. Además, al nosotros ya estar familiarizados con este microcontrolador, ya que es uno de los que se usa en el Laboratorio de Microcontroladores de la ESPOL, el tiempo de desarrollo del proyecto se reduce. Debido a estas razones,

se optó por el uso de este microcontrolador para la implementación del robot.

El uso de motores de pasos en este proyecto se debió principalmente a su mayor precisión con respecto a los motores de corriente directa (DC). Esta precisión se debe a que son controlados de manera digital, por pulsos, y no de manera analógica como los motores DC. Es decir su velocidad siempre será la misma que el usuario desee y no dependerá del voltaje de alimentación. Además, el efecto de frenado en los motores de pasos hace que aumente su precisión, ya que cuando se le envía la orden de detenerse, efectivamente se detienen justo donde se le ordena, a diferencia de los motores DC que luego de haber recibido la orden de frenar, debido a la inercia, se detienen luego de un momento.

Lo motores de pasos son mejores para trabajar a bajas velocidades que los motores DC. Mientras más baja sea su velocidad, más alto será su torque, a diferencia de los motores DC donde ocurre lo contrario. Aunque los motores de pasos no pueden girar tan rápido como los motores DC y no tienen mucho torque cuando aumentan su velocidad, a nuestro robot no le afecta esto, ya que no requiere mayor torque cuando se moviliza a velocidad. Debido a que el robot

debe moverse tanto a bajas como a altas velocidades, se escogieron los motores de pasos para la implementación del robot.

El controlador de motor de pasos que se usa, es la configuración de cuatro MOSFETs, uno por cada bobina [21]. De esta manera, el PIC será el encargado de disparar estos transistores según la secuencia y velocidad requerida. Se prefirió usar el arreglo de los cuatro transistores en lugar de un chip controlador, porque con los transistores se puede manejar motores de mayor corriente. Además al no usar un chip controlador de motor de pasos, resulta más económica la implementación del proyecto. La siguiente figura muestra un controlador de un motor de pasos unipolar, para una de las bobinas.

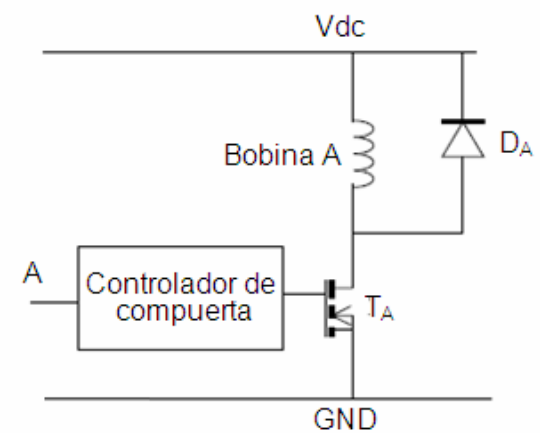

Figura 4.1: Controlador de motor de pasos unipolar

La batería que usa el robot es una de litio-ión. Este tipo de baterías son las de mayor densidad de energía, es decir ofrecen más energía y son livianas. Esto hace que el robot sea más ligero, lo que es una ventaja, ya que podrá moverse con mayor facilidad. Además las baterías de litio-ión tienen la ventaja adicional de mantener su carga cuando no se las utiliza, a diferencia de las de hidruro de níquel-metal que se descargan solas al pasar el tiempo [11].

La batería de litio-ión que usa el robot es de 7.2V y 1200mAh. El voltaje de 7.2V es suficiente para alimentar al regulador de voltaje de la placa del robot. La batería usada es capaz de proveer la corriente que necesita el robot, ya que experimentalmente se determinó que el robot consume menos de 800 mA.

Para suministrar el voltaje requerido por los componentes electrónicos del robot, se usa el regulador de voltaje LM2575-5. Este regulador recibe el voltaje de la batería y lo reduce a los 5V necesarios para energizar el circuito del robot. La ventaja de este regulador sobre el LM7805 es que es mucho más eficiente, es decir disipa menos energía en forma de calor [22]. Gracias a esto la batería durará más tiempo.

4.1.2.2 Transmisor de datos de la aplicación al robot

La interfaz serial fue escogida porque se requieren pocos dispositivos adicionales para armar el circuito transmisor. También porque el microcontrolador ya trae un módulo que interpreta los datos seriales, lo que facilita la comunicación entre la aplicación y el robot.

Debido a que el robot debe ser controlado de manera inalámbrica, se optó por la comunicación por radio frecuencia, la cual no necesita tener línea de vista entre el transmisor y el receptor. Se tomó en cuenta esto, debido a que el robot va a estar moviéndose y girando, además, algo puede interponerse entre el transmisor y el robot. Gracias al uso de la comunicación por RF no se van a presentar problemas de comunicación por falta de línea de vista. Sin embargo cuando hay obstáculos entre el emisor y el receptor, la distancia máxima que puede haber entre ambos disminuye.

Para la comunicación por radiofrecuencia se usan unos módulos transmisor/receptor RF (TLP/RLP-434). Debido a que estos módulos ya vienen listos para ser conectados y usados, se ahorra tiempo de desarrollo. Además, ya que estos módulos ya vienen en una tarjeta pequeña, ocupa poco espacio en la placa del robot.

Los módulos TLP/RLP-434 (transmisor/receptor) fueron escogidos, debido a que en este proyecto se requiere que la comunicación sea unidireccional. Es decir no se justifica el uso de un transmisor-receptor embebidos en un solo módulo *(transceivers)*, ya que estos permiten comunicación bidireccional. Entre las ventajas que tienen los módulos TLP/RLP-434 frente a los *transceivers* SRF-418L, se pueden mencionar su menor costo y menor tamaño.

4.2 Implementación de la aplicación

En general la aplicación realiza 3 tareas que podríamos clasificarlas como una tarea principal y dos secundarias. La tarea principal es interpretar la presencia y/o movimientos de objetos captados de una escena como órdenes que deban enviarse al robot. Mientras que las secundarias son mostrar una retroalimentación de los objetos captados en la escena y brindar la facilidad al usuario de poder configurar las características de los objetos que se deban buscar.

Para poder realizar la tarea principal la aplicación debe estar constantemente capturando la imagen de la cámara web y realizar los cálculos para detectar y seguir los objetos de interés. Sin embargo, si una aplicación es implementada de esta manera no

podrá realizar ninguna otra tarea debido a que esta funcionalidad se lo implementaría en un bucle hasta que la aplicación termine. Para poder implementar lo que hemos propuesto la única manera es desarrollando una aplicación que maneje múltiples hilos.

Nuestra aplicación utiliza dos hilos de ejecución. El hilo principal maneja la interfaz gráfica en general, muestra la retroalimentación necesaria al usuario y brinda posibilidad de configurar los objetos de interés. El segundo hilo lo hemos denominado el “hilo de visión”. El cual es responsable de captar la imagen de la cámara de video y realizar el procesamiento necesario para detectar y seguir los objetos de interés y enviar las órdenes interpretadas al robot.

Los dos hilos de ejecución utilizan las imágenes captadas de la cámara web. El hilo de visión es quien obtiene la imagen y la utiliza para realizar el procesamiento de detección y seguimiento de objetos. Mientras que el hilo principal la utiliza para mostrar una retroalimentación de manera gráfica de los objetos captados en la escena, mostrando en una ventana la imagen captada y encerrando en un círculo los objetos detectados. Además de utilizarla en las ventanas de configuración permitiendo al usuario seleccionar con el

puntero del ratón una región en la imagen capturada que representa el objeto a detectar.

Lo descrito anteriormente podría haber representado un problema de desempeño en la aplicación si no se lo hubiera implementado adecuadamente. El motivo de esto es que se puede presentar lo que se denomina: “condiciones de carrera”. Esto ocurre cuando distintos hilos de ejecución tratan de acceder a una variable compartida (en este caso las imágenes capturadas de la cámara de video) al mismo tiempo. Afortunadamente C++ nos brinda facilidades que nos permitió evitar este tipo de problemas definiendo secciones críticas. A continuación mostramos el pseudocódigo del hilo de visión y explicaremos en detalle las tareas que realiza.

```
procedimiento ThreadVision()
        Declarar imagen, hsv, hue, frame, imagenFiguras
        mientras verdadero hacer
                Entrar en sección crítica frame
                        frame = ObtenerFrame()
                Salir de sección crítica frame
                si imagen = NULL entonces
                        Inicializar imagen, hsv, hue, imagenFiguras
                Entrar en sección crítica frame
                        Copiar frame a imagen
                Salir de sección crítica frame
                si procesarSenalVideo = verdadero entonces
                        hsv = Transformar(imagen, RGB a HSV)
                        hue = ObtenerMatiz(hsv)
                        para cada figura en figuras hacer
                                figura->Buscar(imagen, hsv, hue)
                                figura->SetDataCommRobot()
                        MostrarResultados()
                Entrar en sección crítica imagenFiguras
                        Copiar imagen a imagenFiguras;
                Salir de sección crítica imagenFiguras
                EnviarDatosAlRobot()
        Liberar recursos utilizados
```

Figura 4.2: Pseudocódigo del hilo de visión

En el pseudocódigo podemos ver que existen dos secciones críticas, una llamada frame y la otra imagenFiguras. Esto garantiza que solo un hilo de ejecución pueda utilizar las variables compartidas frame e imagenFigura, evitando que ocurran condiciones de carrera entre nuestros dos hilos.

Tal como lo habíamos descrito anteriormente el hilo principal es el encargado de mostrar la retroalimentación al usuario. Esto lo realiza utilizando las variables compartidas mencionadas. La pantalla que muestra los objetos detectados utiliza la variable imagenFigura ya que en ella se encuentra el resultado del procesamiento realizado. Mientras que la pantalla de configuración de los objetos de interés utiliza la variable frame para permitir al usuario poder seleccionar la región.

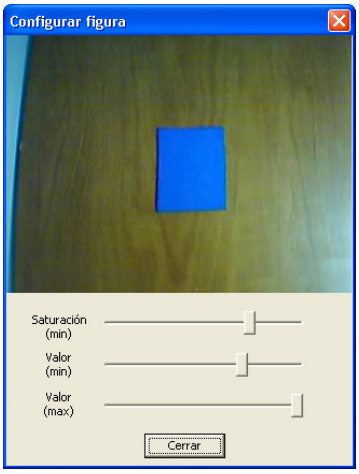

Figura 4.3: Pantalla de configuración de las figuras

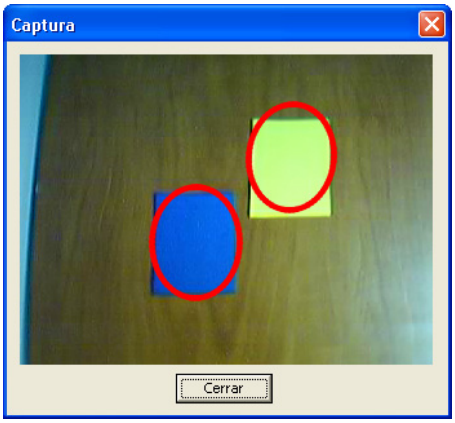

Figura 4.4: Pantalla de resultados

Una vez resuelto este problema podemos enfocarnos en la tarea principal de la aplicación, realizar la detección y seguimiento de los objetos de interés. A continuación mostramos la porción del pseudocódigo del hilo de visión que realiza esta tarea.

```
si procesarSenalVideo = verdadero entonces
        hsv = Transformar(imagen, RGB a HSV)
        hue = ObtenerMatiz(hsv)
        para cada figura en figuras hacer
                figura->Buscar(imagen, hsv, hue)
                figura->SetDataCommRobot()
        MostrarResultados()
```

Figura 4.5: Pseudocódigo del procesamiento de la señal de video

ProcesarSenalVideo es una bandera que utiliza la aplicación para determinar si es que tiene que realizar la detección y seguimiento de los objetos. El programa da la facilidad al usuario de poder iniciar o detener en cualquier momento el procesamiento de la señal de video.

Esto resulta muy útil cuando se están configurando los objetos de interés.

En la iteración de la Figura 4.5 podemos darnos cuenta de la existencia de la variable figuras. Esta variable contiene un arreglo cuyos elementos representan a los objetos de interés. En este arreglo existen 4 elementos: el primero representa al motor izquierdo, el segundo al motor derecho, el tercero al zumbador y por último el cuarto a los frenos. Cada uno de estos elementos contiene las características del color en HSV del objeto de interés al cual representa. Las cuales son configuradas en la pantalla que podemos apreciar en la Figura 4.3.

Los elementos del arreglo figuras son instancias de la clase Figura. Cada una de ellas contienen atributos que permiten almacenar las características del objeto físico configurado además de su estado actual, es decir si es que ha sido detectado o no y si es que lo ha sido su posición. Esta clase también contiene métodos que permite almacenar los valores de los atributos en archivos para que puedan ser usados cuando la aplicación inicie nuevamente sin tener que configurar los objetos de nuevo. Por último contiene un método para

configurar los datos que necesita mandar al robot y un método para buscar el objeto en una imagen dada.

```
class Figura {
        private:
                IplImage *_mask, *_backproject;
                CvBox2D _track_box;
                CvConnectedComp _track_comp;
        public:
                TipoFigura _tipo;
                CvRect _track_window;
                CvHistogram *_hist;
                int _detectado;
                int _smin, _vmin, _vmax;
                int _valor;
                CvScalar _bgr;
        public:
                Figura(TipoFigura tipo);
                ~Figura(void);
                void SetDataCommRobot();
                void Guardar();
                COLORREF GetColor();
                void Buscar(IplImage *img, IplImage *hsv, IplImage *hue);

        private:        void CargarDatos();
};
```

Figura 4.6: Definición de la clase Figura

A continuación mostramos el código del método Buscar de la clase Figura.

```
void Figura::Buscar(IplImage *image, IplImage *hsv, IplImage *hue) {
        if (!_mask)     {
                        _mask = cvCreateImage( cvGetSize(image), 8, 1 );
                        _backproject = cvCreateImage( cvGetSize(image), 8, 1 );
                        _track_window = cvRect(0,0,image->width,image->height);
        }
        cvInRangeS( hsv, cvScalar(0,_smin,_vmin,0), cvScalar(180,256,_vmax,0), _mask);
        cvCalcBackProject( &hue, _backproject, _hist);
        cvAnd( _backproject, _mask, _backproject, 0 );
        cvCamShift( _backproject, _track_window,
                        cvTermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
                        &_track_comp, &_track_box );
        _track_window = _track_comp.rect;
        if( image->origin )
                        _track_box.angle = -_track_box.angle;
        if (_track_box.size.width > 10 && _track_box.size.height > 10) {
                        cvEllipseBox( image, _track_box, CV_RGB(255,0,0), 3, CV_AA, 0 );


                        _detectado = 1;
        }
        else {
                        _track_window = cvRect(0,0,image->width,image->height);
                        _detectado = 0;
```

Figura 4.7: Código fuente del método buscar de la clase Figura

El procedimiento buscar es invocado en el hilo de visión (ver Figura 4.5) por cada uno de los objetos de interés. Recibe como parámetro la imagen capturada de la cámara de video, la imagen transformada del modelo RGB a HSV y por último el componente matiz de la imagen HSV. El algoritmo detecta la posición del objeto de interés utilizando los parámetros especificados y los valores de los atributos que fueron configurados en la pantalla de configuración.

Lo que busca este procedimiento es poder aplicar el algoritmo de desplazamiento medio continuamente adaptativo. Debido a que es un algoritmo que busca el máximo local en un conjunto de datos, lo primero que necesitamos hacer es proveer esos datos de manera adecuada. La información que tenemos de los objetos de interés son características del color. En la pantalla de configuración de cada objeto se obtiene un histograma del componente matiz de los colores en HSV presentes en la región de la imagen seleccionada por el usuario. Adicionalmente tenemos la saturación mínima del objeto y la intensidad mínima y máxima. Todos estos valores fueron configurados a las instancias de la clase Figura que representa cada uno de los objetos de interés.

El procedimiento Buscar prepara los datos para el algoritmo de desplazamiento medio continuamente adaptativo de la siguiente manera. Primero crea una máscara haciendo binaria la imagen en HSV. El criterio utilizado son los componentes de saturación mínima y el rango de intensidad configurado. Es decir que este proceso inicial crea una imagen en blanco y negro cuyos píxeles de color blanco corresponden a los píxeles de la imagen HSV que cumplan con la condición de que sus componentes de saturación sean mayor o igual al valor mínimo de saturación configurado y los valores de intensidad estén entre la intensidad mínima y máxima configurada. Una vez realizado esto se calcula la retroproyección del componente matiz de la imagen HSV utilizando el histograma calculado del objeto de interés. Por último al aplicar una operación AND entre la retroproyección y la máscara tenemos como resultado una imagen en niveles de gris que nos da información de los píxeles que cumplan con las características del objeto de interés. Esta es la imagen resultante que servirá como dato de entrada para el algoritmo de desplazamiento medio continuamente adaptativo.

Figura 4.8: Imagen de la escena antes de la detección

Figura 4.9: Imagen del retroproyección

Figura 4.10: Imagen de la detección de la figura de color azul

Como habíamos descrito anteriormente el algoritmo de desplazamiento medio continuamente adaptativo busca máximos locales en un conjunto de datos. En este caso nuestros datos son los valores de intensidad de los píxeles de una imagen. A medida que los valores de los píxeles sean mayores significa que estos cumplen con más características del objeto de interés. El algoritmo de desplazamiento medio continuamente adaptativo toma las precauciones necesarias para ignorar los picos que se generan por los datos aberrantes presentes en cualquier tipo de imagen. Lo que busca es básicamente la región de la imagen con el mayor número de píxeles de mayor intensidad. Y nos arroja como resultado una recta con esa región que contiene la centroide del objeto de interés.

El algoritmo de desplazamiento medio continuamente adaptativo utiliza como dato de entrada un parámetro adicional que es una recta con la región en donde buscar. Inicialmente el algoritmo recibe una recta con el tamaño total de la imagen. Una vez que haya detectado un objeto en las siguientes iteraciones recibirá la recta que indica la región del objeto detectado.

Después de realizar la búsqueda de la figuras en la imagen se envían los datos al robot para que ejecute la acciones pertinentes.

Para el caso de los motores derecho e izquierdo los objetos siempre deberían estar presentes en la escena y la velocidad de cada uno de ellos es representada por la posición del objeto de interés en la escena. Esto quiere decir que si el objeto esta en el centro de la escena la velocidad es 0. A medida que este más arriba del centro su velocidad se incrementará, caso contrario decrecerá. Para el caso del freno y zumbador el único parámetro a considerar para enviar los datos al robot es la presencia de los objetos que representan dichas acciones. La aplicación constantemente mostrará la retroalimentación necesaria al usuario con las órdenes que se estén enviando al robot.

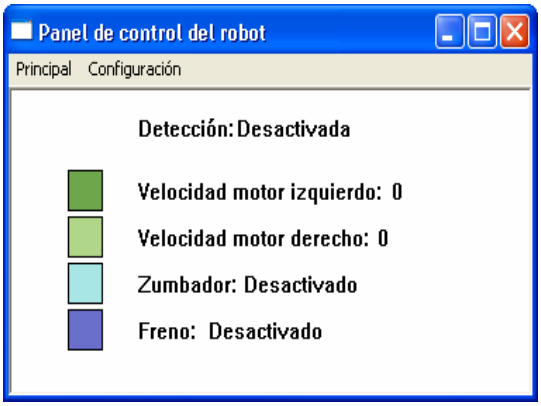

Figura 4.11: Pantalla principal de la aplicación

4.3 Implementación del hardware

4.3.1.1 Robot Móvil

Dentro del robot, el dispositivo que recibe la información enviada por radiofrecuencia, es el receptor RF (RLP-434). Este módulo viene listo para ser conectado directamente a la placa. Una vez que es energizado, empieza a receptar las señales RF. Los datos que recibe este módulo, los cuales provienen del circuito transmisor de datos de la aplicación, a su vez son enviados por medio de un cable, de manera serial, al microcontrolador.

Con el fin de mejorar la recepción de señales RF, el módulo receptor requiere que se le coloque una antena. Esta antena es simplemente un hilo de cable UTP, y va conectada a la entrada correspondiente del módulo. La longitud de la antena es de 16.5cm, la cual corresponde a un cuarto de la longitud de onda que se recibe [23].

El PIC tiene varias entradas y salidas. Una entrada recibe de manera serial los comandos provenientes de la aplicación, por medio del módulo receptor RF. Como salidas del microcontrolador, tenemos las señales que disparan los MOSFETs, además de la señal que

controla al zumbador y una señal que enciende un led que indica cuando el robot está detenido.

Para el correcto funcionamiento del microcontrolador PIC, se le conecta un cristal de 20 MHz, junto con dos capacitares. Esto va a generar la señal de reloj con la cual funcionará el PIC. Al incluir este cristal se ejecutarán las instrucciones del PIC a mayor velocidad. Además el cristal tiene mayor precisión que un oscilador RC [24], lo que es ventajoso para el proyecto implementado, ya que mientras más preciso sea el PIC, más exactos serán los movimientos del robot.

El PIC ejecuta las órdenes que se le envía desde la aplicación a través de los módulos RF. Para esto tiene grabado en su memoria de programa, algoritmos y subrutinas que se encargarán de operar el robot, según la orden que reciba. Estos algoritmos de control se encargarán de generar la secuencia de disparo que necesitan los MOSFETs para mover los motores a la velocidad deseada.

El arreglo de transistores MOSFET (IRF530) conectados al PIC, está compuesto por ocho transistores en total. Se usan cuatro por cada motor. Estos a su vez se conectan a los motores. Estos transistores

son necesarios porque el PIC no puede proveer suficiente corriente para que trabajen los motores.

Entre las órdenes que puede enviar la aplicación al robot, está la de activar o desactivar un zumbador. Esta acción se la realiza por medio de un transistor, cuya base está conectada a una salida del PIC.

Para la alimentación del robot se usa una batería y un regulador de voltaje. Los transistores de los motores, así como la placa electrónica del controlador del robot son energizados por el regulador de voltaje (LM2575-5), el cual recibe el voltaje de la batería y entrega un voltaje de 5V.

En resumen, el robot móvil se compone de: El módulo receptor RF (RLP-434), el microcontrolador (PIC16F877A), los ocho transistores MOSFET (IRF530) que controlan a los motores y los dos motores de pasos que le dan movimiento al robot. La alimentación del robot consta de la batería de Litio-Ion y el regulador de voltaje (LM2575-5)

A continuación se muestra un diagrama de bloques del robot móvil. Además, en las siguientes figuras se muestran fotos del robot implementado.

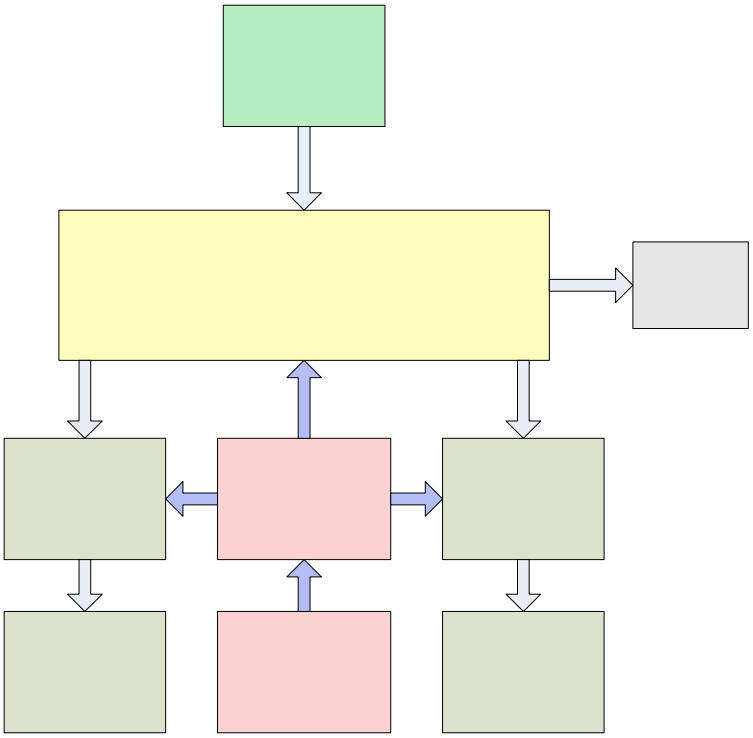

Figura 4.12: Diagrama de bloques del robot móvil

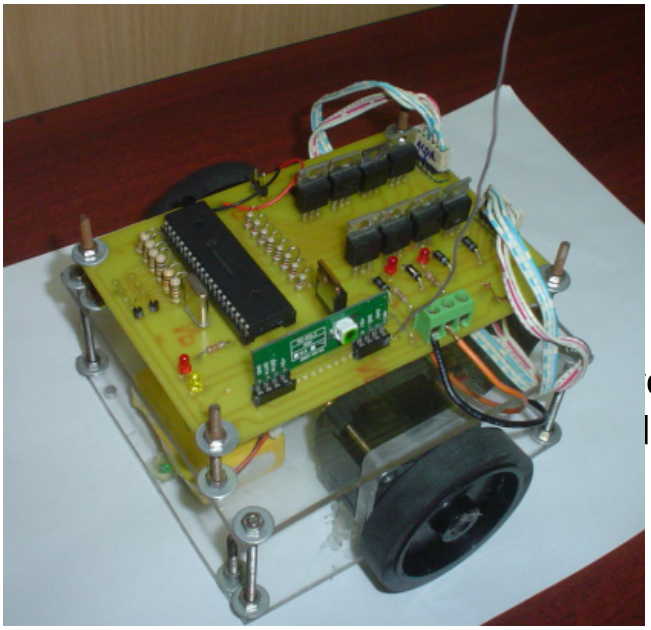

Figura 4.13: Foto 1 del robot móvil

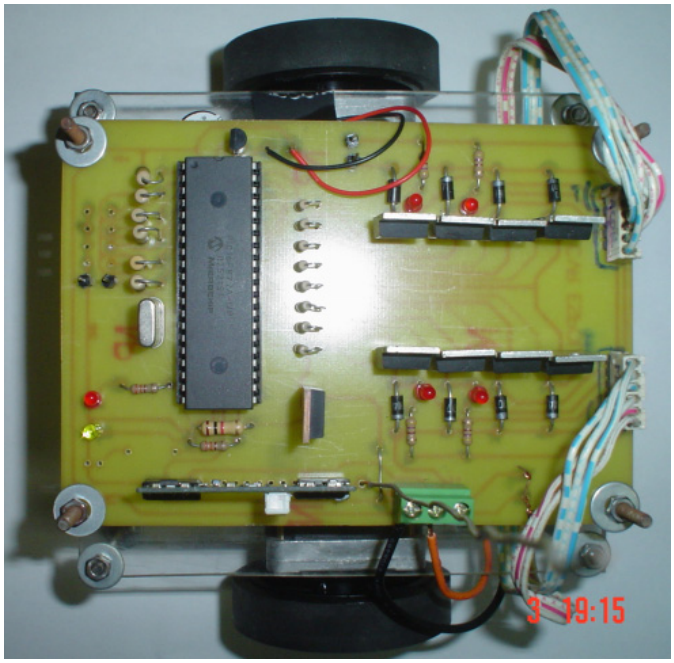

Figura 4.14: Foto 2 del robot móvil

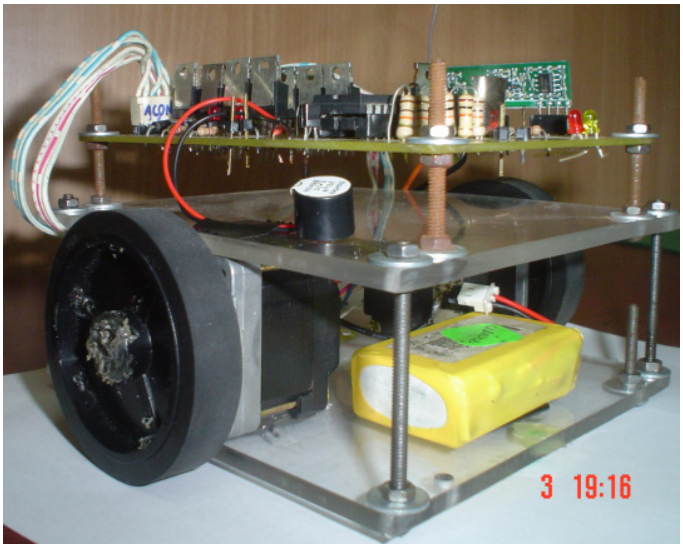

Figura 4.15: Foto 3 del robot móvil

4.3.1.2 Circuito transmisor de datos de la aplicación al robot

El puerto serie del computador es usado como interfaz entre la aplicación y el circuito transmisor. Por esta razón, el circuito transmisor incluye un conector DB-9, desde el cual se conecta un cable serial al computador.

Las señales enviadas por el puerto serie del computador no pueden ser conectadas directamente al módulo transmisor RF. Esto se debe a que ambos operan con diferentes niveles de voltaje. Es necesario incluir un dispositivo que sirva como interfaz entre ambos. Este dispositivo es el DS14C232, el cual convierte los niveles de voltaje del computador a niveles de voltaje TTL. Una vez convertidas las señales a los niveles TTL son enviadas al módulo transmisor RF.

El módulo transmisor RF tiene una entrada digital de datos. Los datos que este recibe, son modulados y transmitidos vía RF. La modulación y transmisión de los datos son realizadas automáticamente por el módulo transmisor RF, es decir no se requiere que el usuario diseñe o configure nada.

El módulo transmisor RF requiere que se le coloque una antena para mejorar la transmisión de los datos. Esta antena es un hilo de cable UTP, la cual se la conecta a la entrada respectiva del módulo. La longitud de la antena del transmisor, al igual que la del receptor, es de 16.5cm, la cual corresponde a un cuarto de la longitud de onda que se transmite.

Para la alimentación del circuito se incluye un conector para una batería de 9V. Además, para tener un voltaje de 5V para poder energizar los componentes del circuito, se usa un regulador de voltaje LM7805. Opcionalmente se puede usar un adaptador de voltaje en lugar de la batería de 9V. Gracias al regulador, no hay restricciones para el uso de un adaptador de voltaje. Se puede usar cualquier adaptador, siempre y cuando la salida de este sea mayor a 7V.

En resumen, el circuito transmisor de datos de la aplicación al robot consta de: El conector del cable serial(DB-9), el convertidor de niveles de voltaje de RS232 a TTL (DS14C232), el módulo de transmisión RF (TLP434) y la antena. Además para la alimentación el circuito, se usa un conector para batería de 9V y un regulador de voltaje(LM7805).

A continuación se muestra un diagrama de bloques del circuito transmisor de datos.

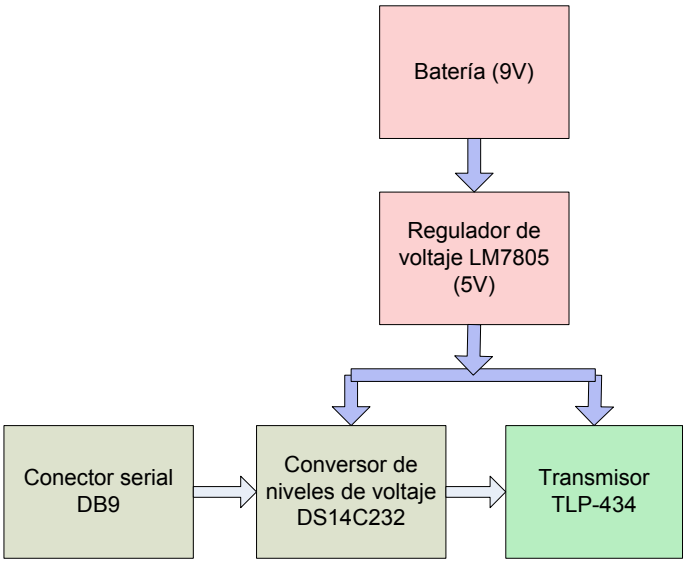

Figura 4.16: Diagrama de bloques del circuito transmisor

CAPÍTULO 5

5 PRUEBAS Y RESULTADOS EXPERIMENTALES

En el presente capítulo se describen las pruebas realizadas al proyecto implementado. Para cada prueba se mencionan los pasos y la metodología a seguir para poder realizarla. Se muestran los resultados obtenidos y las implicaciones que tienen dichos resultados.

5.1 Pruebas del reconocimiento de objetos según su color

En esta prueba se comprobará la eficiencia del algoritmo en la detección de las figuras basándose en información del color. Para esto vamos a detectar 3 figuras de distinto color. Para cada una de ellas mostraremos los datos de la configuración del color, la imagen original, la imagen con la retroproyección y la máscara y por último la imagen resultante. Cabe recalcar que los datos de la configuración del color dependen de la iluminación de la escena en donde se realizaron las pruebas.

Figura	Smin	Vmin	Vmax
Amarilla	0	212	256
Azul	44	188	256
Verde	138	75	173

Tabla 5.1: Valores de la configuración del color de las figuras

Figura 5.1: Imagen inicial de la figura de color amarillo.

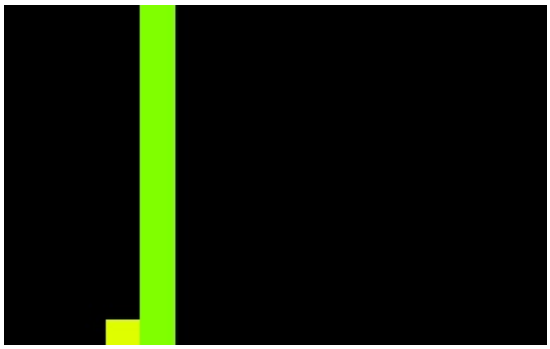

Figura 5.2: Imagen del histograma de la figura de color amarillo.

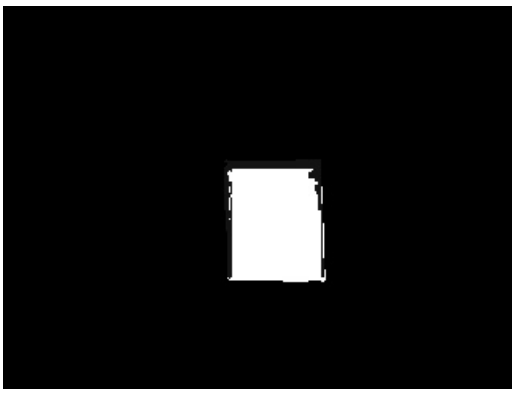

Figura 5.3: Imagen del retroproyección y máscara de la figura de color amarillo.

Figura 5.4: Imagen final con la detección de la figura de color amarillo.

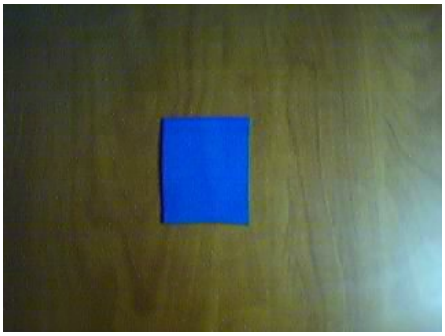

Figura 5.5: Imagen inicial de la figura de color azul.

Figura 5.6: Imagen del histograma de la figura de color azul.

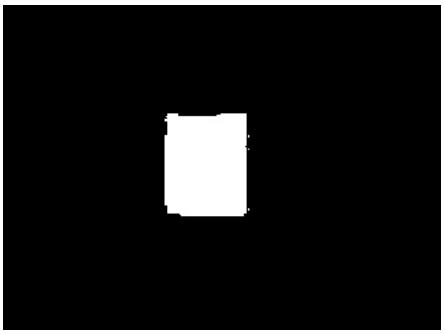

Figura 5.7: Imagen del retroproyección y máscara de la figura de color azul.

Figura 5.8: Imagen final con la detección de la figura de color azul.

Figura 5.9: Imagen inicial de la figura de color verde.

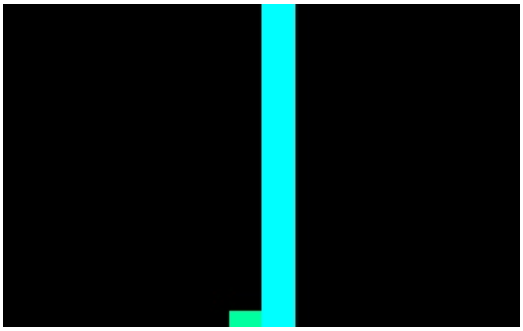

Figura 5.10: Imagen del histograma de la figura de color verde.

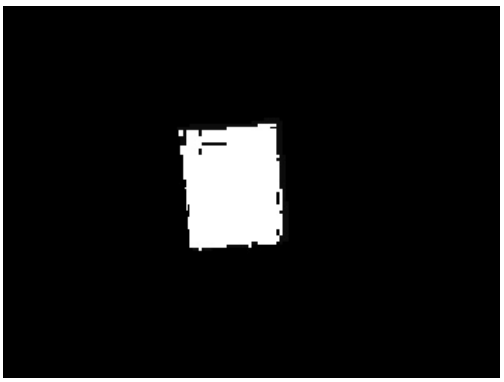

Figura 5.11: Imagen del retroproyección y máscara de la figura de color verde.

Figura 5.12: Imagen final con la detección de la figura de color verde.

Después de realizar las pruebas de detección podemos darnos cuenta de que es un algoritmo de muy buena precisión. Sin embargo como todo algoritmo de visión por computador es necesario contar con las condiciones de iluminación adecuadas.

5.2 Pruebas del seguimiento de objetos

En esta prueba se comprobará la precisión del algoritmo para realizar el seguimiento de los objetos. Debido a que el seguimiento de un objeto esta basado en el cálculo de su posición, necesitamos comprobar la exactitud del algoritmo al obtener la centroide del objeto detectado.

Para la realización de esta prueba vamos a utilizar la configuración de la figura de color azul de la prueba anterior. Se ubicará la figura en la escena captada por la cámara de video y sin moverla dejaremos que la aplicación realice los cálculos de la centroide para 10 iteraciones.

Iteración	Centroide
1	(320,245)
2	(321,244)
3	(322,245)
4	(320,245)
5	(319,244)
6	(320,245)
7	(323,245)
8	(320,244)
9	(320,245)
10	(323,245)

Tabla 5.2: Centroides de la figura de color azul.

Como podemos apreciar en los resultados obtenidos, las variaciones en el cálculo de la centroide son mínimas y suficientes para las necesidades de nuestra aplicación.

5.3 Pruebas de envío y recepción por radiofrecuencia de tramas de datos

En esta prueba se comprobará el funcionamiento real de los módulos de comunicación RF. A continuación se observará el ruido presente en la transmisión. Además se analizarán las señales del transmisor y del receptor.

Para la realización de esta prueba, se conecta la punta de prueba del canal 1 del osciloscopio en el pin de entrada del transmisor. Además, la punta de prueba del canal 2 se conecta al pin de salida del receptor RF.

Una de las principales limitantes de este tipo de módulos es el ruido presente que afecta la transmisión. Este ruido es en parte debido a la modulación ASK que es usada por estos dispositivos, la cual interpreta la variación de la amplitud de la señal portadora como información.

A continuación se muestra una gráfica tomada del osciloscopio del ruido presente a la salida del receptor. En esta gráfica, el canal 1 muestra la señal enviada, en la entrada del transmisor (0V). El canal 2 muestra la señal recibida, en la salida en el receptor. Si no existiera el ruido, esta señal debería ser 0V, como la que envía el transmisor. Sin embargo, debido al ruido, esta señal en realidad es oscilatoria y aleatoria, como lo muestra la figura.

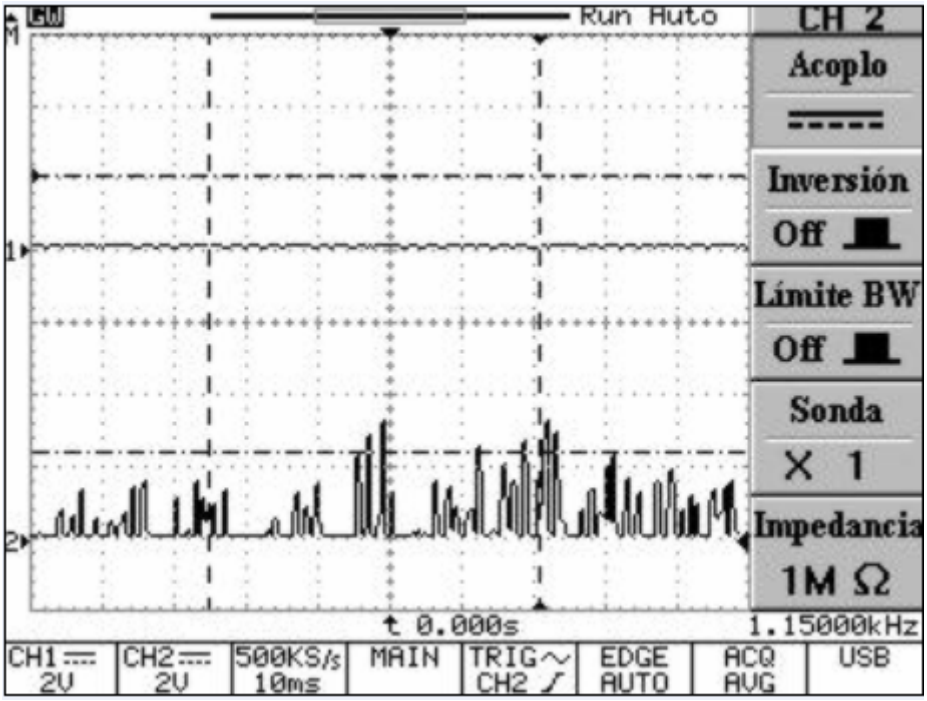

Figura 5.13: Ruido presente en el módulo receptor RF

Para solucionar este problema, se pueden hacer mejoras tanto en hardware como en software. Para el primer caso, se necesita incluir

un filtro a la salida del receptor para bloquear el ruido y permitir que solo pase la señal deseada. Para el caso de mejora en software, es necesario incluir una codificación en cada byte enviado en la trama de datos. De esta forma será menos probable que se trabaje con datos erróneos, ya que cada byte es verificado que contenga el código correcto para que sea considerado como byte válido. Mientras más bits se usen como código identificador, menos probabilidad de que interfiera el ruido habrá. Además siempre es necesario incluir algún método para comprobar errores en la transmisión.

En las pruebas realizadas se comprobó que era suficiente el uso de la codificación y de comprobación de errores para controlar el ruido. En este caso no fue necesario el uso de un filtro.

La siguiente gráfica muestra la señales del transmisor y el receptor simultáneamente. El canal 1 muestra la señal enviada por el transmisor y el canal 2 muestra la señal que sale del receptor.

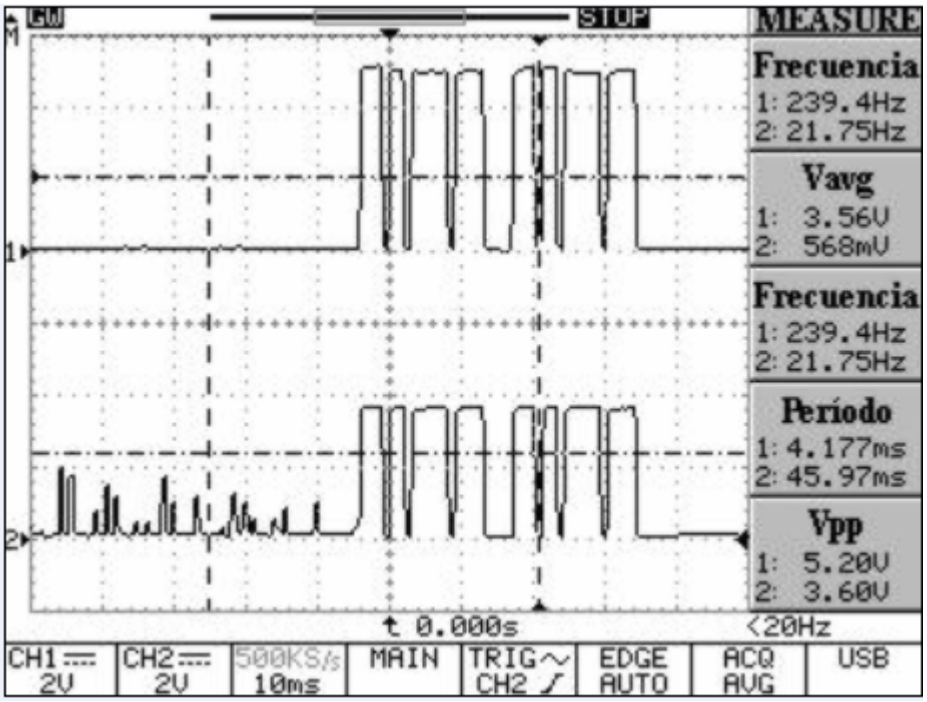

Figura 5.14: Comparación entre la señal enviada y la recibida

La siguiente gráfica muestra el envío y recepción de otra trama de datos. El canal 1 representa la señal enviada por el transmisor y el canal 2 la señal que sale del receptor.

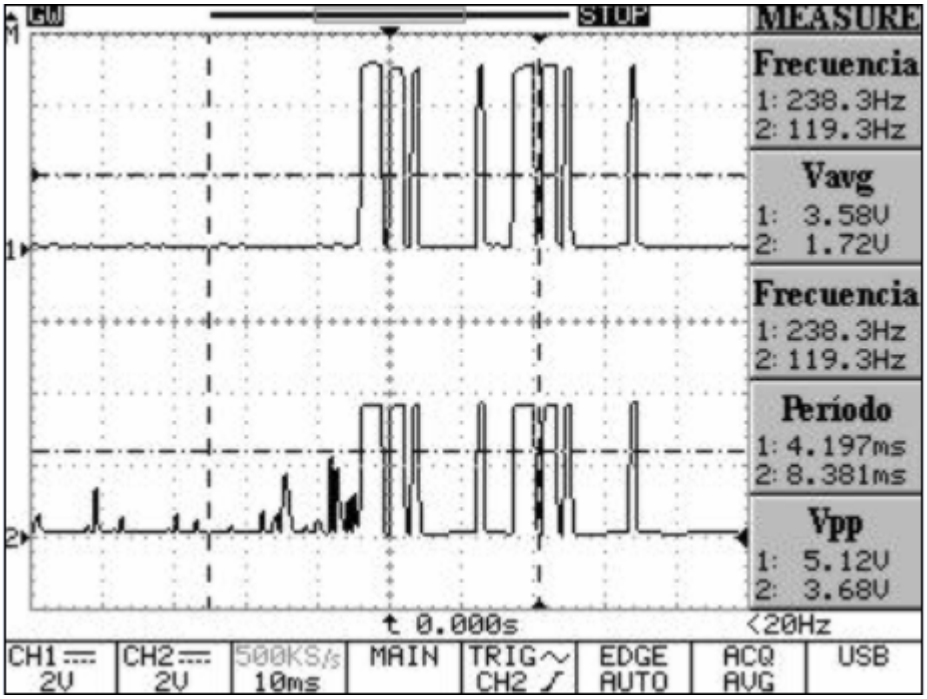

Figura 5.15: Comparación entre la señal enviada y la recibida

Como se puede observar, la señal recibida es igual a la señal enviada mientras dura la trama de datos. Sin embargo se observa una atenuación de la señal, propia de estos dispositivos. Esta atenuación no afecta la interpretación de las señales por parte del microcontrolador, ya que ese nivel de voltaje alto sí es entendido como tal por el PIC.

5.4 Medición de las velocidades del robot

Esta prueba se realiza para determinar experimentalmente las distintas velocidades del robot.

El cálculo de la velocidad, se basa en la ecuación cinemática v=x/t, donde v es la velocidad, x es la distancia recorrida y t es el tiempo que se demora en recorrer esa distancia.

Se hizo recorrer al robot una distancia de 121.7cm. Para cada velocidad se cronometró el tiempo respectivo que el robot se tomó en recorrer dicha distancia. Cada medición de tiempo se realizó dos veces, Luego se obtiene un promedio de las mediciones de tiempo. Para obtener el valor de cada velocidad se divide la distancia de 121.7cm para cada tiempo promedio medido. A continuación se presenta una tabla con los tiempos tomados y la respectiva velocidad obtenida.

Velocidad	Medición t1 (s)	Medición t2 (s)	t prom. (s)	V=x/t (cm/s)
1	35,03	34,74	34,89	3,49
2	27,09	27,26	27,18	4,48
3	21,35	21,67	21,51	5,66
4	16,31	16,1	16,21	7,51
5	10,86	10,82	10,84	11,23
6	6,74	6,76	6,75	18,03
7	4,78	4,69	4,74	25,70

Tabla 5.3: Medición de las velocidades del robot

5.5 Tiempo de respuesta del robot a órdenes del usuario

La siguiente prueba determina el tiempo que transcurre desde que el usuario ordena una acción por medio de la aplicación, hasta que el robot ejecuta dicha acción. Para esto consideramos el retardo que hay en la transmisión RF, así como también el tiempo que se toma el computador y el microcontrolador en procesar dichas órdenes.

Entonces el retardo total es la suma de tres tiempos: T=Tc+Tt+Tm, donde T es el retardo total desde que el usuario realiza una orden hasta que el robot la ejecuta. Tc es el tiempo que se toma el computador en procesar dicha acción y enviar el comando correspondiente por el puerto serie. Tt es el retardo que hay en la transmisión RF, desde que la señal sale del emisor hasta que llega al

receptor. Tm el tiempo que se toma el microcontrolador desde que recibe el comando del módulo RF, hasta que finalmente ejecuta la acción correspondiente.

Para determinar el tiempo que se toma el computador en procesar las órdenes del usuario vamos a considerar el peor de los casos, cuando no hay ningún objeto de interés presente en la escena. Esto se debe a que el algoritmo de desplazamiento medio continuamente adaptativo hace seguimiento a los objetos, es decir que en la iteración i + 1 busca el objeto cerca del área en donde fue detectado en la iteración i. Al no haber ninguna figura presente en la escena en todas las iteraciones el algoritmo realizará la búsqueda en toda la imagen.

Para poder calcular este tiempo vamos a tener que modificar un poco el código del hilo de visión para que antes y después de realizar el procesamiento imprima la hora actual. Sacando la diferencia tendremos el tiempo que se demora la aplicación en hacer la detección en el peor de los casos.

Muestra	Tiempo en milisegundos
1	16
2	16
3	15
4	15
5	16
6	16
7	16
8	16
	Promedio = 15.75 ms

Tabla 5.4: Medición del tiempo de interpretación de órdenes del usuario

Ahora determinaremos el retardo que existe en el envío de las señales de un módulo RF a otro. La siguiente figura nos ayuda para tal propósito. Esta gráfica de osciloscopio cuyo canal 1 muestra lo enviado por el transmisor y el canal 2 la señal que sale del receptor, es útil para determinar el desfase o retardo que existe entre ambas señales.

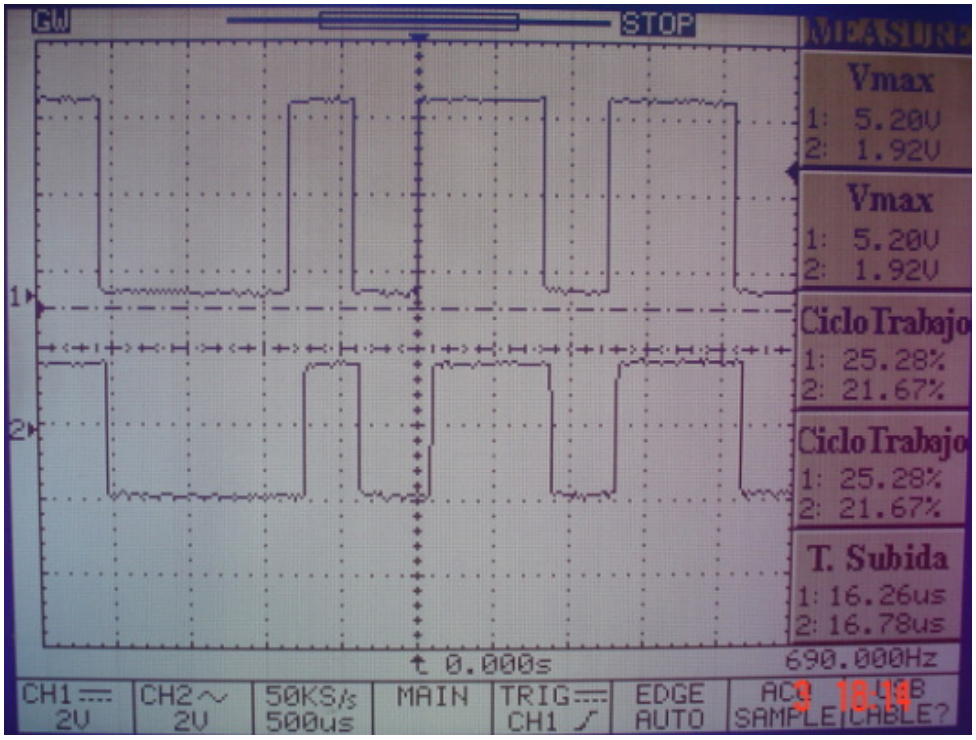

Figura 5.16: Retardo en la transmisión RF

Como vemos en la figura anterior, la diferencia entre ambas señales es de 0.2 cuadros. La escala de tiempo del osciloscopio corresponde a que cada cuadro es igual a 500us. Entonces, el desfase entre ambas señales es 0.2*500us. Por lo tanto, el retardo desde que la señal es enviada por el emisor, hasta que sale del receptor es de 100us.

Finalmente, el tiempo que se toma el microcontrolador en procesar las órdenes del usuario se lo determina de la siguiente manera: Primero determinamos mediante el osciloscopio el tiempo que dura

toda la trama de datos. Como vemos en la siguiente gráfica, el tiempo total de la trama es aproximadamente 33ms.

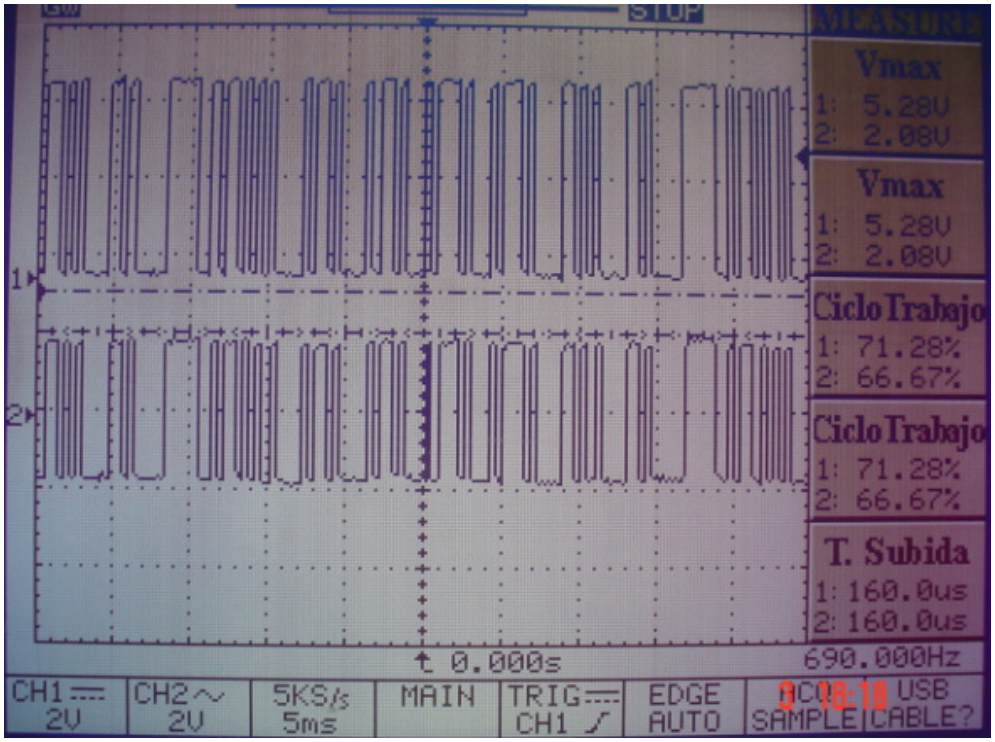

Figura 5.17: Duración de la Trama de datos enviada

Debido a que el PIC trabaja con un cristal de 20MHz, cada instrucción la realiza en 0.2us. El número total de líneas de código del programa escrito en C es alrededor de 70. Las cuales son traducidas a lenguaje ensamblador por el compilador de C. Considerando que cada línea de código en C produce cinco líneas de código en ensamblador, tenemos 350 instrucciones. Lo que en tiempo equivale a 350*0.2us que es igual a 70us.

Estos 70us que se demora el PIC en ejecutar las instrucciones son despreciables comparados con los 33ms que se demora en leer toda la trama de datos. Por lo tanto se puede considerar que el tiempo que se demora el PIC en ejecutar las órdenes enviadas por el usuario es Tm = 33ms.

Una vez que tenemos los tres tiempos, para determinar el retardo total, simplemente los sumamos. T = Tc+Tt+Tm = 15.75ms +100us+33ms, lo que nos da que el retardo total desde que el usuario ordena una acción hasta que el robot la ejecuta es 48.85ms. Como vemos este tiempo es sumamente pequeño, tanto así que en realidad el retardo es imperceptible para el usuario.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. El uso de motores de pasos en la implementación del robot incrementa la precisión del mismo. Esto se debe a que este tipo de motores funciona con pulsos, cada pulso equivale a una cantidad determinada de grados de rotación. Incluso si la batería se empieza a descargar el motor no variará su velocidad, ya que esta no depende del voltaje de alimentación, como el caso de los motores de corriente continua. Además, la precisión en el frenado del robot también es precisa, ya que cuando se le envía la orden de paro, los motores se enclavan en la posición actual.

2. En la prueba experimental donde se medían las velocidades del robot se pudo verificar la precisión del mismo. Se tomaron dos mediciones para cada velocidad, como se pudo comprobar, ambas mediciones fueron bastante parecidas. Esto muestra claramente la confiabilidad en la precisión del robot, tanto en posición como en velocidad.

3. El algoritmo de detección y seguimiento de objetos mostró buenos resultados en las pruebas experimentales. La precisión y velocidad del algoritmo fueron las adecuadas para las necesidades de nuestra aplicación.

4. La transmisión RF es vulnerable al ruido, esto es perjudicial para cualquier tipo de aplicación. En nuestro caso fue necesario el uso de una codificación de cada byte y de implementar un método para la verificación de errores de la transmisión. Una vez implementado este protocolo se comprobó experimentalmente que la transmisión no introduce errores en el control del robot.

5. El tiempo de respuesta del robot a las órdenes enviadas por el usuario puede considerarse como instantáneo para este tipo de aplicaciones. Esto es válido porque el retardo que existe entre la orden del usuario y la ejecución de dicha orden por el robot es sumamente pequeño, del orden de los milisegundos.

6. Después de realizar todas las pruebas necesarias podemos concluir que hemos cumplido con los objetivos que nos hemos propuesto en esta investigación. El algoritmo de detección y seguimiento de objetos es lo suficientemente preciso y rápido para cumplir con las demandas exigentes de una aplicación en tiempo real. Así mismo logramos implementar un robot que interprete adecuadamente las órdenes que envía el usuario a través del movimiento de objetos.

RECOMENDACIONES

1. En toda aplicación de visión por computador hay que tener en consideración la cantidad de memoria RAM que consume la aplicación. Un descuido por parte del programador podría ocasionar la sobreutilización de este recurso trayendo consigo problemas de rendimiento en el computador que ejecute el programa.

2. Los requerimientos de este proyecto tuvieron como consecuencia la implementación de una aplicación que maneje múltiples hilos de ejecución y variables compartidas entre estos hilos. El acceso y escritura de estas variables por cada hilo ocasiona el problema denominado “condiciones de carrera”. Esto se puede evidenciar como un incremento exagerado en el porcentaje de uso de CPU ocasionado por la aplicación. Para evitar esto definimos en nuestro código secciones críticas asegurando de esta manera que un solo hilo de ejecución a la vez pueda hacer uso de las variables compartidas.

3. El uso de módulos de radio frecuencia para control a distancia y monitoreo es ampliamente utilizado en la actualidad. Siempre hay que tener en cuenta las características de la aplicación en particular que se vaya a implementar con estos dispositivos para poderlos elegir

correctamente. Los parámetros más importantes que se deberían considerar son: La distancia máxima, el tipo de modulación, la frecuencia a la que transmite, la potencia de salida, la velocidad máxima de transmisión y el voltaje de alimentación.

FUTUROS TRABAJOS

Los resultados obtenidos en este proyecto podrán ser utilizados en cualquier aplicación de visión por computador que utilice como criterio de detección y seguimiento de objetos información del color. Brinda una alternativa más eficiente y precisa comparándolo con los métodos tradicionales que utilizan el modelo de color RGB. Al utilizar el algoritmo de desplazamiento medio continuamente adaptativo garantizamos un mejor método de detección ya que se da un tratamiento estadístico a los datos descartando el ruido presente en la imagen.

En el área de robótica podemos destacar el uso de un algoritmo sencillo para controlar a los motores de paso, ya que no fue necesario el uso de otro integrado sino que se lo hace directamente desde el PIC. Por otra parte, la implementación de la codificación para la transmisión de datos con los módulos de radio frecuencia TLP434 para poder prescindir de los filtros hechos mediante hardware, fue de gran importancia. Estas contribuciones serán de mucha utilidad para futuros proyectos en el área de control a distancia, control de motores de paso y electrónica en general.

APÉNDICES

A APÉNDICE A: CIRCUITOS ESQUEMÁTICOS

A.1 Robot Móvil

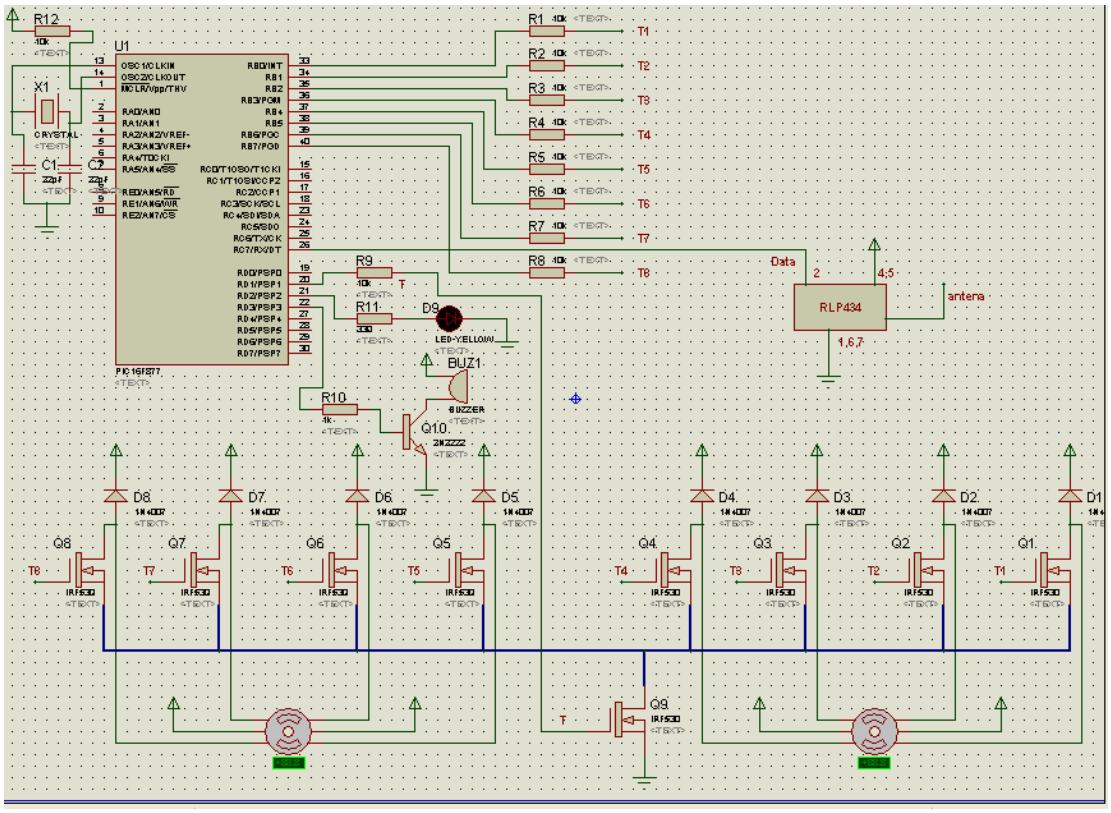

Figura A.1: Circuito esquemático del robot móvil

A.2 Fuente de Poder del Robot Móvil

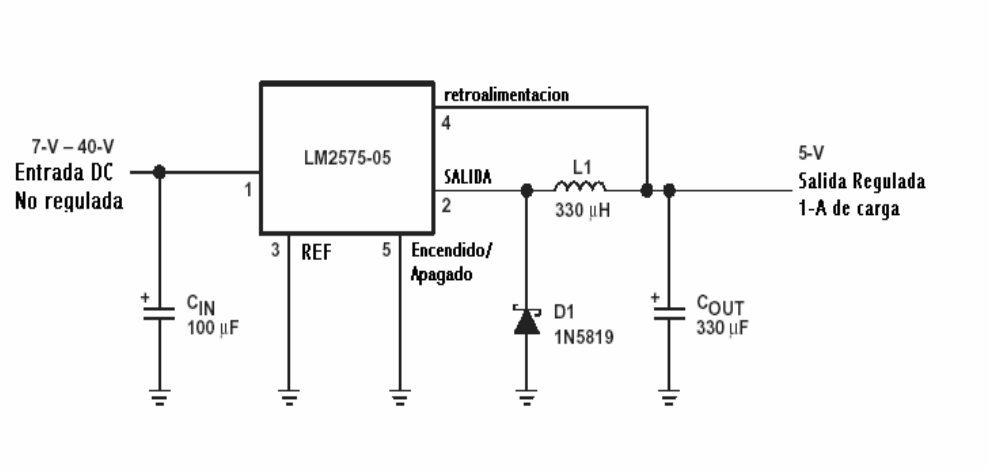

Figura A.2: Circuito esquemático de la fuente de poder del robot móvil

A.3 Transmisor de datos de la aplicación al robot

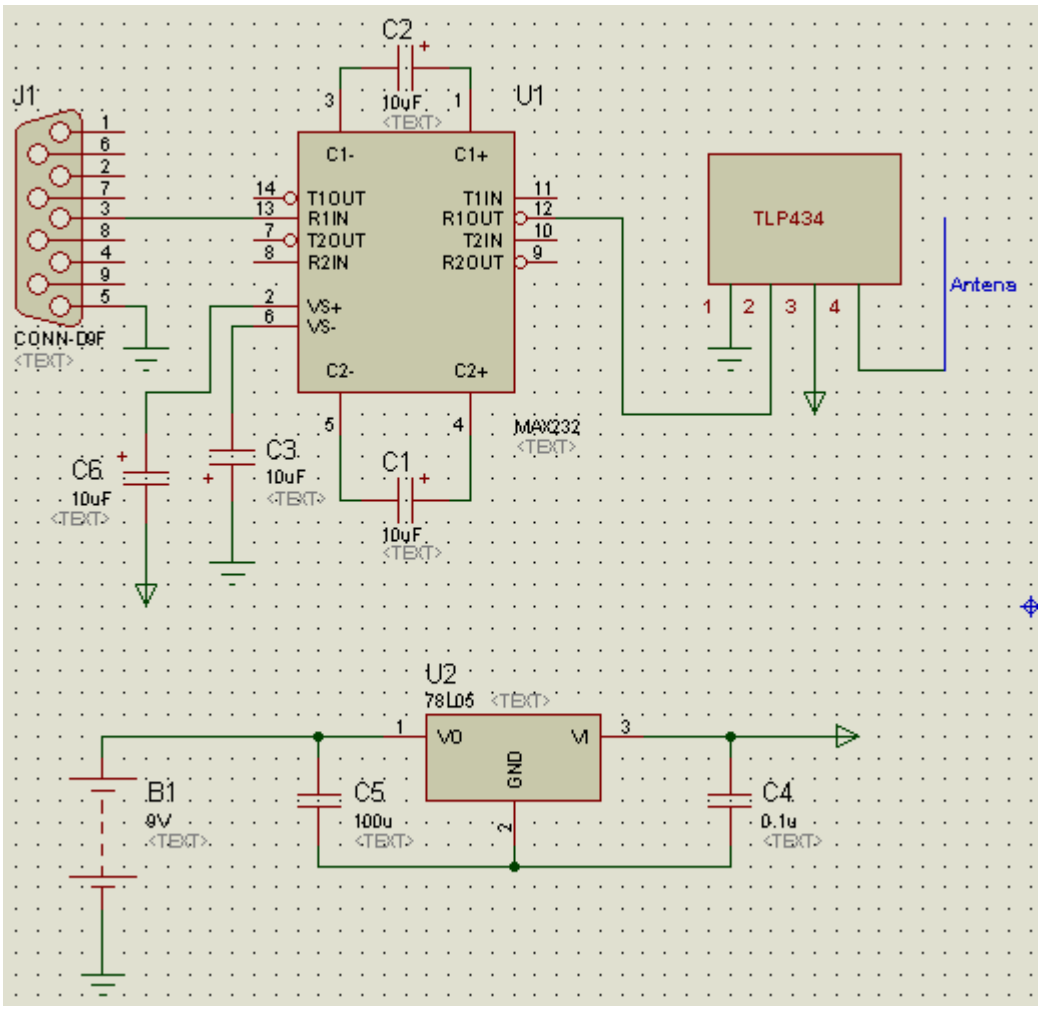

Figura A.3: Circuito esquemático del transmisor

B APÉNDICE B: CIRCUITOS IMPRESOS

B.1 Robot Móvil

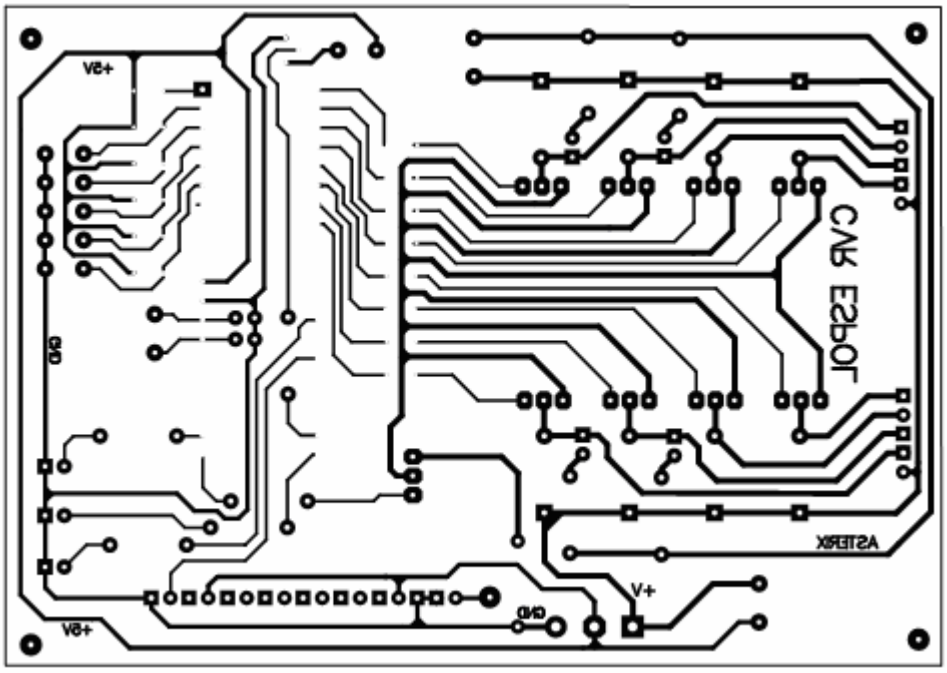

Figura B.1: Circuito impreso del robot móvil

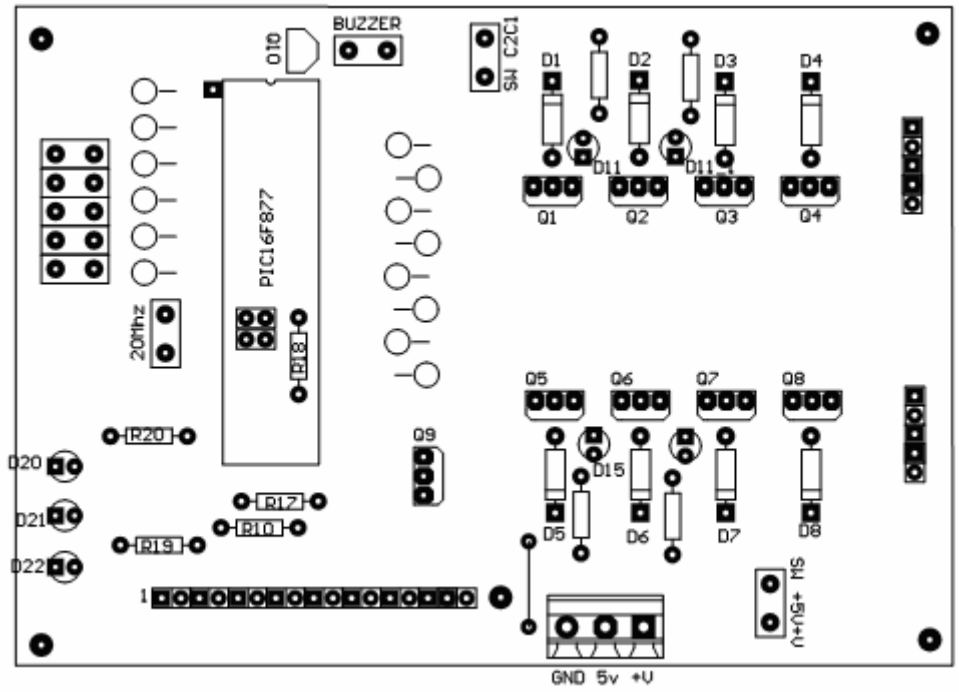

Figura B.2: Componentes en la placa del robot móvil

B.2 Fuente de Poder del Robot Móvil

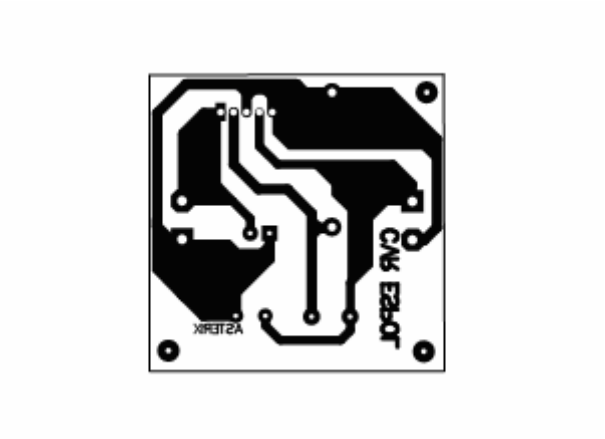

Figura B.3: Circuito impreso de la fuente de poder del robot móvil

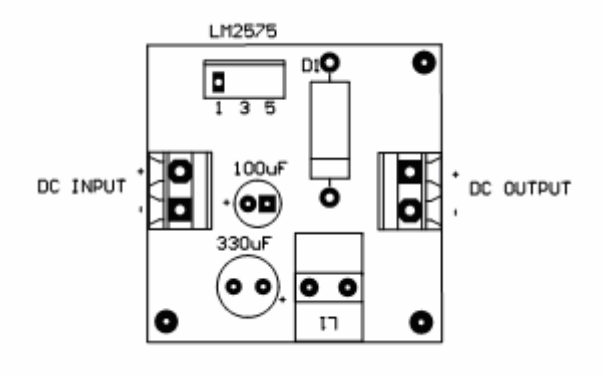

Figura B.4: Componentes en la placa de la fuente de poder del robot móvil

B.3 Transmisor de datos de la aplicación al robot

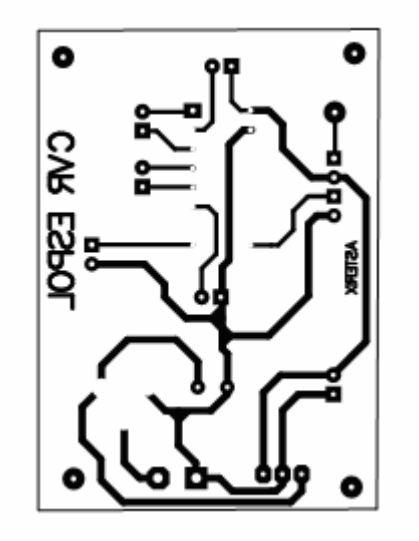

Figura B.5: Circuito impreso del transmisor

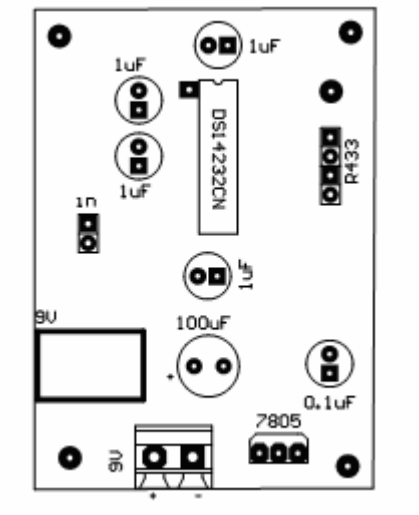

Figura B.6: Componentes en la placa del transmisor

C APÉNDICE C: CÓDIGO FUENTE DEL MICROCONTROLADOR PIC16F877A

/* Código fuente escrito en el IDE del compilador Pcwh de la compañía CCS */

```
#include <16F877A.h>
#device adc=8
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
#use delay(clock=20000000)
#use rs232(baud=2400,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)

#define cero 8
#define Bvf 90     //Byte de verificación de la transmisión
#define NB 6       //número de bytes de la trama, después de Bvf
int1 f1, f2, f3, f4;
int1 dira, dirb;
int mota, motb;
signed k, i, m;
int out[4];
int codein[NB];
int c1, c2, max1, max2, t, t1, v1, v2, ac;
int vel1, vel2, code, codigo;
int1ft;
int1 mot, ac3, ac4, motp, mot1;
#bit d1=8.1
#bit d2=8.2
#bit d3=8.3
```

```
#BYTE portb = 6
#BYTE pc = 7
#BYTE pd = 8
#BYTE trisc = 0X87
#BYTE trisd = 0X88
#BYTE trisb = 0X86

#int_RTCC
RTCC_isr() {
  if(!mot1&(v1!=cero)){    // generador de secuencia motor A
    c1--;
    if(c1==0){
      c1=max1;
      if(dira==0){
        k--;
        if(k<0)  k=3;
      }
      if(dira==1){
        k++;
        if(k>3)  k=0;
      }
    }
  }
  mota=out[k];
  portb=mota+motb*16; // motA->portB(4lsb), motB->portB(4msb)
}

#int_TIMER2
TIMER2_isr() {
  if(!mot1&(v2!=cero)){    // generador de secuencia motor B
```

```
      c2--;
      if(c2==0){
        c2=max2;
        if(dirb==0){
          i--;
          if(i<0)  i=3;
        }
        if(dirb==1){
          i++;
          if(i>3)  i=0;
        }
      }
    }
    motb=out[i];
    portb=mota+motb*16; // motA->portB(4lsb), motB->portB(4msb)
}

#int_TIMER1
TIMER1_isr() {
  t++;
}

void serial();
void velconv();
void indic();
void paromot();
void acciones();

void main() {
  int start, time, time2;
```

```
int p, cont;
setup_adc_ports(NO_ANALOGS);
setup_adc(ADC_OFF);
setup_psp(PSP_DISABLED);
setup_spi(FALSE);
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_4);
setup_timer_1(T1_INTERNAL|T1_DIV_BY_4);
setup_timer_2(T2_DIV_BY_4,255,1);
setup_comparator(NC_NC_NC_NC);
setup_vref(FALSE);
enable_interrupts(INT_RTCC);
enable_interrupts(INT_TIMER1);
enable_interrupts(INT_TIMER2);
enable_interrupts(GLOBAL);
set_timer0(0);
set_timer2(0);
trisa=0xfe;
trisb=0;
k=0; i=0;
out[0]=8;
out[1]=4;
out[2]=2;
out[3]=1;
v1=cero; v2=cero;
c1=5; c2=5; t=2;
max1=c1; max2=c2;
trisd=0; trisc=0x80 ;
pd=0; pc=0;
ac=0;
do {
```

```
    serial();
    velconv();
    acciones();
    paromot();
    d1=(!mot1);  // maneja al mosfet que desconecta los motores
    d2=mot1;  // indicador de que los motores están desconectados
    d3=ac3;  //  zumbador
  } while (TRUE);
}

void serial(){  // entrada serial de datos provenientes del receptor RF
  code=getc();
  if(code==Bvf){
    for(m=0;m<NB;m++)
      codein[m]=getc();
    for(m=0;m<(NB-1);m=m+2){
      if(codein[m]==codein[m+1]){
        codigo=codein[m]/16;
        if(codigo==10)
          v1=codein[m]&15;
        if(codigo==13)
          v2=codein[m]&15;
        if(codigo==6)
          ac=codein[m]&15;
      }
    }
  }
}

void velconv(){     //cambio referencia de velocidad v1 y v2
```

```
if(v1<=cero){
   vel1=cero-v1;
   dira=0;
}
else{
   vel1=v1-cero;
   dira=1;
}
if(v2<cero){
   vel2=cero-v2;
   dirb=1;
}
else{
   vel2=v2-cero;
   dirb=0;
}
// tabla de conversión de velocidad
if(vel1==0) max1=255;
if(vel1==1) max1=255;
if(vel1==2) max1=200;
if(vel1==3) max1=160;
if(vel1==4) max1=120;
if(vel1==5) max1=80;
if(vel1==6) max1=50;
if(vel1==7) max1=35;
if(vel2==0) max2=255;
if(vel2==1) max2=255;
if(vel2==2) max2=200;
if(vel2==3) max2=160;
if(vel2==4) max2=120;
```

```
    if(vel2==5) max2=80;
    if(vel2==6) max2=50;
    if(vel2==7) max2=35;
}

void acciones(){  // acciones: activar zumbador, detener motores
  if(BIT_TEST(ac,1))   mot=1;  // acción detener motores
  else  mot=0;
  if(BIT_TEST(ac,2))   ac3=1;  // acción activar zumbador
  else  ac3=0;
}

void paromot(){   //desconectar motores si están parados
  if(((v1==cero)&(v2==cero))){
    if(ft==0)
      t1=t+5;
    ft=1;
    if(t1<=t)
      motp=1;
  }
  else{
    ft=0;
    motp=0;
  }
  mot1=mot|motp;
}
```

D APÉNDICE D: CÓDIGO FUENTE DE LA CLASE FIGURA DE LA APLICACIÓN

```
/*
 *  Archivo: figura.cpp
 *  ====================
 *  Contiene la implementación de la clase figura que representa
 *  a los objetos de interés a buscar.
 *
 *  Autor: Danilo Matamoros
 */

#include <stdio.h>
#include "figura.h"
#include "CommRobot.h"

extern DataCommRobot dataEnvio;

/*
 *  Constructor: Figura
 *  ====================
 *  Incializa los atributos del objeto con los valores que
 *  fueron configurados la última vez que se ejecutó la aplicación
 */
Figura::Figura(TipoFigura tipo) {
   _mask = 0;
   _tipo = tipo;
   _detectado = 0;
   _valor = 0;
   _bgr.val[0] = 255.00;
   _bgr.val[1] = 255.00;
   _bgr.val[2] = 255.00;
   _smin = 0;
   _vmin = 0;
   _vmax = 256;
   CargarDatos();
}
```

```
/*
 *  Procedimiento: CargarDatos
 *  ==========================
 *  Carga los datos configurados desde un fichero.
 */
void Figura::CargarDatos() {
    FILE *fp;
    char nombreArchivo[24];
    // se carga el histograma
    sprintf(nombreArchivo, "%d_hist.xml", _tipo);
    _hist = (CvHistogram *) cvLoad(nombreArchivo);
    // se carga lo demás
    sprintf(nombreArchivo, "%d.dat", _tipo);
    fp = fopen(nombreArchivo, "r");
    if (fp == NULL)
        return;
    int b, g, r;
    fscanf(fp, "(%d,%d,%d),%d,%d,%d", &b, &g, &r,&_smin, &_vmin,
                        &_vmax);
    _bgr.val[0] = (double)b;
    _bgr.val[1] = (double)g;
    _bgr.val[2] = (double)r;
    fclose(fp);
}

/*
 *  Procedimiento: Guardar
 *  ======================
 *  Guarda los datos configurados a un fichero.
 */
void Figura::Guardar() {
    FILE *fp;
    char filename[24];
    // se guarda el histograma
    sprintf(filename, "%d_hist.xml", _tipo);
    cvSave( filename, _hist);
    // se guarda la demás info
    sprintf(filename, "%d.dat", _tipo);
    fp = fopen(filename, "w");
    fprintf(fp, "(%d,%d,%d),%d,%d,%d", (int)_bgr.val[0],
            (int)_bgr.val[1], (int)_bgr.val[2], _smin, _vmin, _vmax);
    fclose(fp);
}
```

```
/*
 *  Función: GetColor
 *  ==================
 *  Devuelve el color configurado de la figura
 */
COLORREF Figura::GetColor() {
   return RGB( (int)_bgr.val[2],
               (int)_bgr.val[1],
               (int)_bgr.val[0] );
}

/*
 *  Procedimiento: Buscar
 *  =====================
 *  Busca la figura configurada en la imagen que recibe como
 *  parámetro.
 *  image -> imagen orignal en rgb
 *  hsv -> imagen original en HSV
 *  hue -> componente matiz de la imagen en HSV
 *  En caso de que se encuentre la figura configurada se marca con
 *  un círculo  rojo su posición en la imagen orignal en RGB.
 */
void Figura::Buscar(IplImage *image, IplImage *hsv, IplImage *hue) {
   if (!_mask)    {
      _mask = cvCreateImage( cvGetSize(image), 8, 1 );
      _backproject = cvCreateImage( cvGetSize(image), 8, 1 );
      _track_window = cvRect(0,0,image->width,image->height);
   }
   cvInRangeS( hsv, cvScalar(0,_smin,_vmin,0),
               cvScalar(180,256,_vmax,0),
               _mask);
   cvCalcBackProject( &hue, _backproject, _hist);
   cvAnd( _backproject, _mask, _backproject, 0 );
   cvCamShift( _backproject, _track_window, cvTermCriteria(
         CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
         &_track_comp, &_track_box );
   _track_window = _track_comp.rect;
   if( image->origin )
      _track_box.angle = -_track_box.angle;
   if (_track_box.size.width > 10 && _track_box.size.height > 10) {
      cvEllipseBox( image, _track_box, CV_RGB(255,0,0), 3, CV_AA,
                    0 );
      _detectado = 1;
   }
```

```
    else {
        _track_window = cvRect(0,0,image->width,image->height);
        _detectado = 0;
    }
    _valor = (int)(_track_box.center.y / 16);
}

/*
 *  Función: SetDataCommRobot
 *  ========================
 *  Configura los datos a enviar al robot.
 */
void Figura::SetDataCommRobot() {
    switch(_tipo) {
        case MOTOR_DER:
            dataEnvio.velDer = _valor;
            break;
        case MOTOR_IZQ:
            dataEnvio.velIzq = _valor;
            break;
        case ZUMBADOR:
            dataEnvio.accionZumbar = _detectado ? 4 : 0;
            break;
        case DETENER:
            dataEnvio.accionDetener = _detectado ? 2 : 0;
            break;
        default:
            break;
    }
}
```

E APÉNDICE E: MANUAL DE USUARIO DE LA APLICACIÓN

La aplicación esta compuesta de 3 pantallas:

1. La pantalla principal en donde se muestra el estado actual de la aplicación y las órdenes que se están enviando al robot.

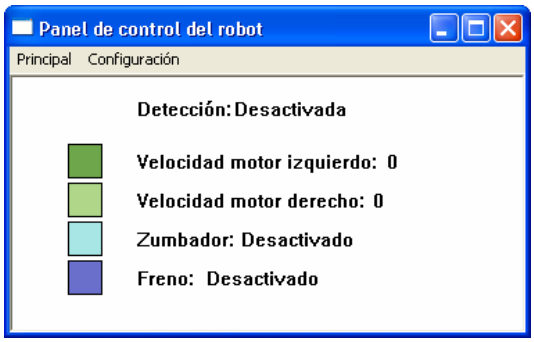

Figura E.1: Pantalla principal

Adicionalmente contiene 2 menús con los siguientes ítems:

- Principal:
  - Iniciar detección: Inicia la detección de los objetos de interés.

- Detener detección: Detiene la detección de los objetos de interés.
- Mostrar resultados: Muestra la pantalla de resultados.
- Salir: Salir de la aplicación

- Configuración: En este menú se muestra un listado de los objetos de interés que se pueden configurar. Al presionar cualquiera de ellos se mostrará la pantalla de configuración de dicho objeto.

2. La pantalla de resultados donde se muestra la captura de la cámara de video con los objetos de interés detectados marcados con rojo.

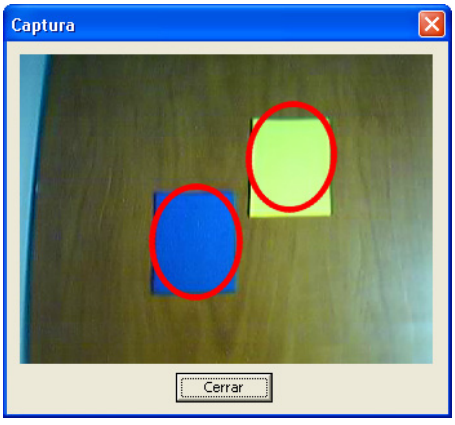

Figura E.2: Pantalla de resultados

3. La pantalla de configuración donde se configura las características de color de cada objeto de interés.

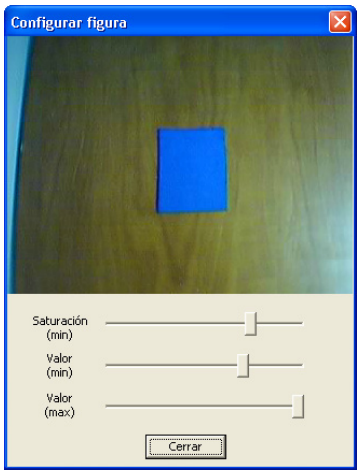

Figura E.3: Pantalla de configuración

Para la configuración el primer paso es seleccionar un área en la imagen que contenga el objeto de interés. Una vez seleccionado se deben ajustar los valores de saturación y valor según lo requiera el objeto. El programa encerrará con un círculo de color rojo los objetos que cumplan con las características configuradas en la imagen.

BIBLIOGRAFÍA

[1] Ishii, H. and Ullmer, B., “Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms” in Proceedings of Conference on Human Factors in Computing Systems CHI '97, ACM Press,1997, pp. 234-235.

[2] Johnson, N., “Learning Object Behaviour Models”, Ph.D. Thesis, University of Leeds, School of Computer Studies, 1998.

[3] Moore, D., “Vision-Based Recognition of Actions using Context”, Ph.D. Thesis, Georgia Institute of Technology, 2000, pp. 8-9.

[4] Casals, A., “Robots de servicios y personales: Aspectos tecnológicos que dificultan su progreso” Artículo Técnico, Universidad Politécnica de Cataluña, Centro de Investigación en Ingeniería Biomédica, 2005, pp. 1-2.

[5] John G. Allen, Richard Y. D. Xu, Jesse S. Jin, “Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces” University of Sydney, School of Information Technologies, 2004

[6] Murcia, J. “Puerto Serie RS-232”, 2006. Disponible en http://www.depeca.uah.es/alcabot/seminario2006/Trabajos/JoseManuelMurciaBarba.pdf

[7] Simonett y Fuhrken SA, “Ampere News”, 2007. Disponible en http://www.ampere.com.mx/news.php?news_no=8

[8] Miles, P. and Carroll, T., *Build Your Own Combat Robot*, McGraw Hill/Osborne, 2006, pp. 167.

[9] Paillacho, D. y Salamea, V., “Diseño e Implementación de un Equipo de Robots Autónomos con Decisiones en Tiempo Real: Futbol Robótico – Componente Electromecánico”, Tesis, Escuela Superior Politécnica del Litoral, Facultad de Ingeniería en Electricidad y Computación, 2003, pp. 203.

[10] Ash, D., “Comparison between OOK/ASK and FSK Modulation Techniques for Radio Links”, RF Monolithics Inc., 2002, pp. 1-3.

[11] McComb, G., *Robot Builder´s Bonanza*, 2nd Edition, McGraw Hill, 2001, pp. 191,192, 283.

[12] TodoRobot, “Tutorial animado sobre Motores Paso a Paso”, 2007. Disponible en http://www.todorobot.com.ar/

[13] Universidad de Alicante, “Sensores externos”, 2007. Disponible en http://www.dccia.ua.es/dccia/inf/asignaturas/ROB/optativos/Sensores/externos.html

[14] Carnegie Mellon University, “Sensor de contacto”, 2007. Disponible en http://www-education.rec.ri.cmu.edu/webpage/touchsensor.htm

[15] Wikimedia Foundation, “Zumbador”, 2007. Disponible en http://es.wikipedia.org/wiki/Zumbador

[16] Ing. Gustavo Martin, “Información Técnica sobre baterías de Litio-Ion”, 2006. Disponible en http://www.dbup.com.ar/li-ion.htm

[17] Fundación EROSKI, “Recomendaciones para el buen uso de las baterías”, 2007. Disponible en http://www.consumer.es/web/es/tecnologia/hardware/2005/06/08/142782.php

[18] Liu, K., “Mobile, Flexible-Link, and Parallel-Link Robots”, *Robotics*, *Mechanical Engineering Handbook*, Crc Press LLC, 1999, pp.14/192.

[19] Iovine, J., *PIC Robotics*, McGraw Hill, 2004, pp. 1.

[20] “PIC16F87XA Data Sheet”, Microchip Technology Inc., 2003, pp. 1.

[21] Rashid, M., *Power Electronics Handbook*, Academic Press, 2001, pp. 714-715.

[22] “LM2575 Data Sheet”, Motorola Inc., 1999, pp. 1.

[23] “AM-RT4-xxx, AM-RT5-xxx User Guide”, ABACOM Technologies, Inc., 2006, pp. 2-3.

[24] “PIC16F87XA Data Sheet”, Microchip Technology Inc., 2003, pp. 145-146.