UNIVERSIDAD MICHOACANA DE SAN NICOLAS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

ROBOT MÓVIL AUTONOMO SEGUIDOR DE LINEA CON RASPBERRY PI Y BIBLIOTECA DE VISIÓN COMPUTACIONAL (OPENCV)

Tesis para obtener el Grado de Ingeniero en Computación que presenta

Héctor Ortega Reyes

Asesor

Moisés García Villanueva

Maestro en Ciencias

Morelia Michoacán, septiembre 2015

Contenido

Lista de Figuras

Lista de Tablas

Resumen

El presente trabajo nos muestra el prototipo de un robot móvil autónomo seguidor de línea mediante visión computacional. Se utiliza la microcomputadora Raspberry Pi como unidad de procesamiento del vídeo para identificar la línea, para ello se emplea la librería de código abierto del área de Visión Computacional (OpenCV, por sus siglas en inglés de Open Computer Vision). Para el control de velocidad del robot se utiliza un control Proporcional Integral Derivativo implementado en una placa Arduino, que además realiza la comunicación serial con la Raspberry Pi a través del puerto USB, recibiendo como parámetro de referencia la información de posición de la línea identificada por el sistema de visión, respecto del eje x.

De las dificultades que se enfrentan en el prototipo esencialmente son cómo identificar y realizar el seguimiento del objeto en cuestión (en nuestro caso la línea), principalmente considerando que se encuentra en movimiento. En este trabajo se presenta además una estrategia de identificación de la línea en forma robusta, señalando la posición central (valor entero en píxeles) del objeto identificado, el cual sirve como parámetro en los datos de información de posición que requiere el control de velocidad del robot móvil.

Se realizaron pruebas para diferentes resoluciones de captura de imágenes, en dos ambientes diferentes de trayectorias y a dos niveles de alimentación de voltaje de los motores.

Los resultados obtenidos en la parte experimental del prototipo nos señalan una velocidad de 14 centímetros por segundo en promedio en los ambientes de prueba.

En relación a la velocidad de procesamiento del vídeo en la microcomputadora Raspberry Pi, se logró almacenar en promedio 5 frames o marcos por segundo en la resolución de 160x140 píxeles, es decir la capacidad de procesamiento para esta tarea.

Palabras clave: Robot, visión, seguidor, móvil, línea.

Abstract

This work presents a prototype of a mobile robot line follower by computer vision. It use the microcomputer Raspberry Pi as the video processing unit to identify the line, it goal is reached trough of the libraries Open source Computer Vision (OpenCV, for its acronym in English). For speed control's robot is used a Proportional Integral Derivative control implemented on Arduino board, which also carries out serial communication with Raspberry Pi microcomputer via USB port, getting information from the robot's position relative to the line that it want to follow, considering only axis x as data of reference.

The challenges faced in the prototype essentially is how to identify and track the object in question (in our case the line), since it is in motion. Also, in this work is presented a robust strategy to identify the line, pointing to the central position (integer value in pixels) of the identified object, which serves as position information parameter for the speed's control of the robot.

We tested for different resolutions of capturing images in two trajectories and two levels of voltaje to supply motor power.

The results obtained in the experimental part of the prototype designed, shows the average speed of 14 centimeter per second in test enviroments.

With respect to processing speed of the video by Raspaberry Pi microcomputer, in average 5 frames per second was stored in the resolution of 160x140 pixeles, i.e. capacity processing under this conditions for the robot's work.

Capítulo 1

Introducción

Un robot es un artificio mecánico capaz de actuar de forma autónoma a la hora de resolver un problema. Existe una multitud de robots diseñados para cumplir diferentes objetivos, brazos robóticos para el montaje de piezas en una fábrica, vehículos aéreos no tripulados(UAV), por mencionar algunos. El esquema general de un sistema robot se resume en lo siguiente:

- Sensores externos que captan una percepción del entorno: Visión, tacto, audición, proximidad, etc.
- Sensores internos que miden el estado de la estructura mecánica: giros, desplazamientos, velocidades, etc.
- Actuadores: Sistemas de control que aseguran el funcionamiento correcto de los movimientos, trayectorias, etc.

Dentro de los robots móviles terrestres, cabe mencionar los vehículos con ruedas, ya que suponen la solución inmediata debido a la eficiencia y simplicidad a la hora de trabajar sobre terrenos duros y llanos. Existe una gran variedad de posibilidades a la hora de diseñar un sistema con ruedas [Baturone01], entre las más famosas se encuentran:

- Configuración Ackerman. Es la configuración que estamos habituados a ver en los vehículos convencionales, por lo tanto es una configuración muy probada y estable. Se

basa en una estructura de cuatro ruedas colocadas en dos ejes, donde solamente las dos ruedas delanteras permiten el giro sobre el eje. Ver figura 1.1.

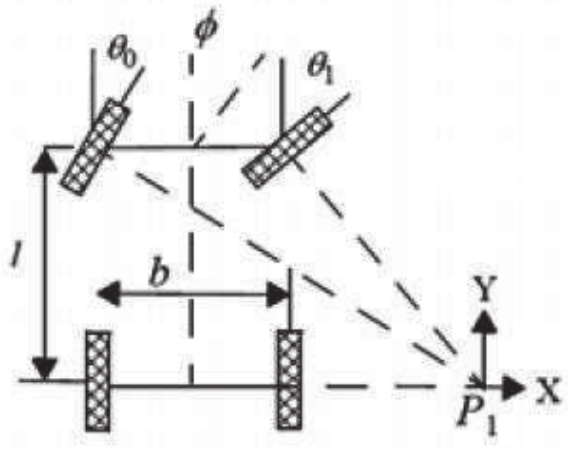


Figura 1.1: Configuración Ackerman de los robots móviles terrestres [xatacaciencia].

- Configuración triciclo. Posee un gran parecido a la configuración Ackerman pero aporta mayor simplicidad en la construcción, ya que solamente se necesitan tres ruedas. No obstante, aporta una menor estabilidad al sistema. Ver la figura 1.2.

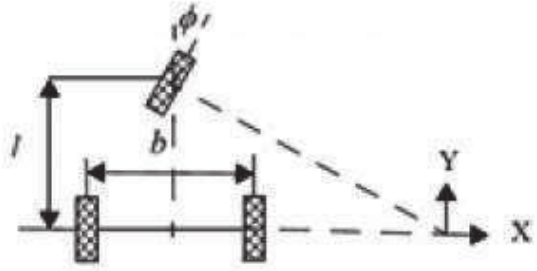


Figura 1.2: Configuración Triciclo para los robots móviles terrestres [xatacaciencia].

- Pistas de deslizamiento. Esta configuración es la que habitualmente se observa en los tanques de guerra. Se les conoce como vehículos tipo oruga y se basan en el direccionamiento por diferencia de velocidades. De esta forma, cuando se requiere realizar un giro, se aumenta la velocidad de una de las pistas para producir el giro en el vehículo. Esta configuración esta pensada para trabajar en terrenos irregulares y

normalmente se suelen incluir ruedas (skid- steering), ver la figura 1.3.

Figura 1.3: Robot militar TALON [globalsegurity.org].

Los robots son utilizados en la industria para realizar trabajos que al hombre le son muy difíciles de realizar o que se necesita más de una persona para realizar cierto trabajo, lo que implica realizar un mayor gasto en personal. Lo que se busca en las empresas es realizar trabajos que impliquen un menor costo pero sin perder la calidad de éstos. Es por esto que los robots son muy útiles en la actualidad.

En las empresas de gran reconocimiento, se puede observar la actividad y trabajo de robots, los cuales son máquinas hechas por el hombre para realizar actividades de una manera más sencilla o realizar completamente las tareas humanas de mayor riesgo, peligro y esfuerzo, tareas que para el hombre resultan difíciles, cansadas e incluso mortales al realizarlas, en aquellos trabajos donde se necesitan varias personas y que se desea mantener o incrementar la calidad de sus productos o servicios que se realizan.

Proveer a los robots de la percepción visual es una tarea que el área de Visión Computacional desarrolla. Los sistemas de visión computacional son una herramienta que se implementa con frecuencia en robots móviles para ayudar en la identificación de objetos, empleando técnicas del área de procesamiento de imágenes se hace el seguimiento de ciertas características en una imagen. La obtención de la información primaria en visión computacional se realiza mediante un dispositivo óptico, una cámara, que simula la percepción

visual del ojo humano.

1.1. Planteamiento del problema

El objetivo en el problema de seguimiento de una trayectoria es trasladar un vehículo a su destino de una forma segura. Para que un vehículo se pueda desenvolver de manera eficaz en diferentes entornos, se debe tener la capacidad de reacción ante situaciones inesperadas que puedan presentarse durante el seguimiento de la trayectoria.

En este trabajo se describe la implementación de un prototipo de robot móvil con un sistema de visión computacional para la detección y seguimiento de una línea trazada en una determinada área.

Se tiene como propósito diseñar e implementar un software para un prototipo móvil controlado por computadora a través de las librerías de OpenCV, la cual es una herramienta de código abierto que nos permite el procesamiento digital de imágenes e implementación de los algoritmos de visión computacional. El prototipo será capaz de detectar, rastrear y seguir un patrón de línea a través de la incorporación de una cámara web que actuará como sensor de las propiedades del objeto a seguir, en este caso una línea negra.

1.2. Justificación

El avance tecnológico hoy y siempre ha hecho necesario realizar la implementación y desarrollo de equipos para la investigación o la industria. En la robótica se requiere de velocidad de procesamiento para implementar la percepción visual y de algoritmos que identifiquen correctamente el objeto de interés.

El procesamiento de imágenes está siendo muy utilizado en los desarrollos tecnológicos de la actualidad. El área de aplicaciones de análisis de imágenes hacen que esta tecnología sea una de las más atractivas, además de que ha sido de gran utilidad y se obtiene un beneficio muy grande en algunas áreas como la robótica, la industria, por mencionar algunas y que algunos años atrás no era viable realizarlas para la mayoría de las personas debido a que requerían un mayor poder de cómputo y aunado a que se realizaban median-

te sensores que son poco accesibles para adquirirlos, ya que el precio es elevado. Con el surgimiento de cámaras económicas y desarrollo de software para manejarlas, se convierte en una ventaja usar una cámara web como sensor al proporcionar flexibilidad a la hora de reconfigurar el software para su posterior uso. Otra ventaja que se encuentra al usar una cámara como sensor es en la economía, ya que es más económico obtener una cámara que un sensor para ciertas aplicaciones. Finalmente es de importancia mencionar que no es muy difícil la programación con las herramientas de software que se tienen en la actualidad.

1.3. Objetivos de la Tesis

1.3.1. Objetivo general

El objetivo de este trabajo es realizar un prototipo de un robot seguidor de línea y diseñar un algoritmo para optimizar el procesamiento realizado en un sistema de visión computacional para la detección de la línea, de tal forma que le permita al prototipo realizar su tarea en forma más eficiente.

1.3.2. Objetivos particulares

- Ensamblar el prototipo del robot seguidor de línea.
- Manejar la librería de visión Computacional OpenCV.
- Implementar la comunicación entre el software (Sistema de Visión) y el hardware del robot seguidor de línea.
- Implementar diferentes técnicas en la detección de la línea.
- Realizar la experimentación y comparación de los resultados.

1.4. Descripción de capítulos

En el capítulo 1 se da una breve introducción a este trabajo. Se plantea el problema a resolver, se describe el objetivo general así como los objetivos particulares de esta tesis.

En el capítulo 2 se introducen algunos antecedentes del área de visión por computadora. Se hará mención de algunas cosas que se pueden realizar con visión computacional aplicada en la robótica además de la aplicación que tienen los robots seguidores de línea en la industria.

En el capítulo 3 describe la librería de OpenCV, que es la utilizada en la realización de esta tesis, además de las ventajas y desventajas de los sistemas de visión computacional.

En el capítulo 4 se hará mención y se describirán las características del hardware que se utilizó para el armado del robot móvil autónomo seguidor de línea.

En el capítulo 5 se realizará la descripción del software implementado para que el robot pueda seguir la línea previamente trazada.

En el capítulo 6 se mostrará la experimentación realizada después de implementar el robot y el software para la tarea a resolver por el robot, así como los resultados obtenidos en las pruebas realizadas.

En el capítulo 7 se presentan las conclusiones de los resultados obtenidos, así como aportaciones y trabajos futuros.

Capítulo 2

Antecedentes

El desarrollo en la tecnología donde se incluyen las poderosas computadoras electrónicas, los actuadores de control con retroalimentación, transmisión de potencia a través de engranes, y la tecnología en sensores han contribuido a flexibilizar los mecanismos autómatas para desempeñar tareas dentro de la industria. Son varios los factores que intervienen para que se desarrollaran lo primeros robots en la década de lo 50's. La investigación en inteligencia artificial desarrolló maneras de emular el procesamiento de información humana con computadoras electrónicas e inventó una variedad de mecanismos para probar sus teorías.

En la actualidad se tiene un concepto de un robot con apariencia humana y que actúa como tal. Este concepto humanoide ha sido inspirado y estimulado por algunas narraciones de ciencia ficción.

El hombre ha ido buscando la forma de ir perfeccionando y evolucionando herramientas para realizar tareas de una manera mas cómoda y eficaz. Se han creado sistemas para operar robots de una manera automática. La tendencia de un trabajo que automatiza uno o varios procesos, es que el ser humano sea el encargado solamente de supervisar el sistema. Las tareas restantes se realizan sin que el hombre intervenga, en su lugar lo llevan a cabo mecanismos o robots que trabajan de una manera más rápida y eficiente.

Por otra parte, el sistema de percepción en un robot que facilita la información del mundo real para que el robot pueda interpretarla, es totalmente dependiente del tipo y

cantidad de sensores que se dispongan, ello determina en gran medida las capacidades del robot.

El uso de los sensores ayuda al problema de navegación en un ambiente de un robot, en el que se persigue el buen manejo de los siguientes puntos:

- Posición actual y Destino.
- Minimizar daños a él mismo (evitar colisionar).
- Minimizar daños al ambiente.
- Máxima eficiencia (energía utilizada vs distancia recorrida).
- Conocer con que objetos se puede interactuar.
- Aprender acerca de su ambiente de operación.

En este trabajo, se tiene como sensor una cámara, la cual es muy útil para la identificación de objetos captando la intensidad luminosa de los mismos. En muchos proyectos de visión artificial (visión computacional), las cámaras tienen la función de sensores tal y como se propone en este proyecto.

2.1. Aplicaciones de visión computacional en robótica

En la actualidad, la visión por computadora algunas veces no es la mejor solución a un problema, en algunas ocasiones, el problema es tan complejo que la solución humana es la mejor, por ejemplo, la conducción de un vehículo en una carretera con trafico intenso. Pero en algunas ocasiones, las soluciones humanas suelen ser inexactas o subjetivas y en ocasiones son a la vez lentas. El sistema de visión humana puede describir automáticamente una textura a detalle, un borde, un color, una representación bidimensional de una tridimensional, ya que puede diferenciar entre imágenes de diferentes personas, firmas, colores, entre otras, además de que puede vigilar ciertas zonas, diagnosticar enfermedades a partir de radiografías, etc. Aunque algunas de estas tareas pueden llevarse a cabo mediante visión artificial, el software o el hardware necesario no consigue los resultados que serían los deseables. A pesar de las limitaciones expuestas, cada día es mayor el número de aplicaciones

de visión computacional, por lo que es muy difícil mencionar todas ellas, sin embargo se hará mención de algunas. Es importante diferenciar entre las aplicaciones donde la visión computacional constituye una herramienta por si sola y aquellas en las que es parte de un sistema multisensorial. El primer caso engloba todas aquellas aplicaciones en las que el único sensor presente es el de visión. En el segundo caso se hace referencia a la navegación en robótica donde la visión constituye una capacidad sensorial más para la percepción del entorno que rodea al robot.

2.1.1. Navegación en robótica

En el caso de la navegación en la robótica, la visión por computadora es un elemento de un sistema conformado por varios sensores. La información que procede de la visión es validada, comparada y finalmente integrada con el resto de la información proporcionada por otro tipo de sensores dando como resultado la reconstrucción de la escena 3D que permite la navegación autónoma del sistema. El uso de la visión por computadora no es exclusivamente para el uso en la robótica, sino que podría utilizarse en otras aplicaciones tales como guiado automático de máquinas.

2.1.2. Biología, geología y meteorología

En el campo de la biología se podrían distinguir entre aplicaciones microscópicas y macroscópicas. En una imagen microscópica nos podemos encontrar con abundantes organismos que, mediante técnicas de segmentación orientadas a regiones podrían ser aislados para su identificación mediante sus propiedades como el tamaño, color, forma u otra característica importante que pueda discriminar los organismos, o también para contar el número de microorganismos o células presentes en una imagen. En las imágenes macroscópicas se pueden utilizar las regiones para la identificación de determinados tipos de texturas en vegetales o características de diferentes áreas naturales por su color o el crecimiento de ciertas especies. En geología se puede también detectar movimientos de terrenos captando dos imágenes en diferentes momentos de tiempo para observar la variación mediante una diferencia de imágenes que tengan condiciones similares de iluminación. En meteorología se podrían utilizar las técnicas de detección y predicción del movimiento para observar la

evolución de ciertas masas nubosas u otros fenómenos meteorológicos a través de imágenes recibidas vía satélite.

2.1.3. Medicina

En el área de medicina se tienen muchas aplicaciones en las que aparece el procesamiento de imágenes, muy frecuentemente orientadas hacia el diagnóstico de dolencias o enfermedades entre las que se incluyen las radiografías, resonancias magnéticas, tomografías, etcétera.

2.1.4. Identificación de construcciones, infraestructuras y objetos en escenas de exterior

Mediante imágenes aéreas o satelitales se puede determinar la presencia de ciertas regiones a través de la segmentación de las mismas así como detectar la presencia de ciertas construcciones como edificios, o infraestructuras como carreteras, canales o puentes a través de algunas técnicas de extracción de bordes o contornos combinadas con la segmentación de regiones.

2.1.5. Reconocimiento y clasificación

Otra aplicación que tiene la visión por computadora es la clasificación de objetos por su tamaño y además el recuento de los mismos, por ejemplo el caso de contar monedas con respecto del área de la moneda o perímetro, el conteo de personas en el acceso a un espacio físico, en un empaque de productos contar el número de objetos que deben incluirse en el contenedor.

2.1.6. Inspección y control de calidad

La inspección de un objeto manufacturado puede tomar muchas formas que podría involucrar algunas tareas como las que se mencionan a continuación:

- Verificar la presencia de cada característica esperada.

- Verificar las dimensiones de esas características por ejemplo radio y longitud de un cilindro.
- Verificar las interrelaciones entre características como centros de gravedad y ángulos entre planos por mencionar algunos.

La inspección se refiere a la verificación de si un objeto cumple con determinados criterios, lo que implica comparar el objeto con algún objeto modelo que describa las características más relevantes del objeto. En algunas ocasiones existen tolerancias definidas dentro de las cuales las medidas realizadas pueden considerarse como aceptables dentro de la información obtenida. Un objetivo de un sistema de control de calidad consiste en detener la producción de algún producto si el sistema de producción comienza a generar productos que no cumplen con las normas estándares generales. En algunos procesos industriales de producción se requiere que los productos manufacturados sean inspeccionados para asegurar que algunas medidas de calidad y fiabilidad se cumplan. Algunas características con las que debe cumplir cualquier sistema de inspección basado en visión son los siguientes:

- Flexibilidad para poder inspeccionar cualquier tipo de objetos.
- Resolución para que el sistema sea capaz de proporcionar medidas que sean suficientemente precisas para poder establecer si los objetos inspeccionados están dentro de la tolerancia.
- Resolución espacial para decidir si las características inspeccionadas están posicionadas y orientadas con la presición adecuada con respecto a un sistema de referencia.
- Facilidad de uso para que su manipulación y versatilidad no sea compleja.

En muchas ocasiones las técnicas que se desarrollan en los sistemas no satisfacen estos requisitos, algunas razones se mencionan a continuación:

- Son muy sensibles a los pequeños cambios de iluminación y a la reflectancia de las superficies.
- Sólo trabajan bien en superficies con textura uniforme.

- Es difícil deducir las profundidades absolutas de los puntos.
- La resolución de profundidad o tercera dimensión obtenida resulta insuficiente.

En la inspección visual geométrica es muy importante considerar el hecho de que los objetos geométricos a inspeccionar van a admitir una serie de valores de tolerancia en las medidas de los elementos geométricos que componen a los objetos.

En muchos casos es más sencillo inspeccionar objetos utilizando un perfil o una serie de perfiles de las diferentes vistas del objeto a inspeccionar. En este caso, los problemas de adquisición tridimensional son eliminados y las medidas se hacen con referencia a los datos de forma bidimensional. Hay que tener cierto cuidado en que la posición de la cámara y las condiciones de iluminación sean idénticas con el fin de que las vistas del objeto sean las mismas en cada imagen y en cuanto a la cámara, hay que especificar por mencionar algo, que la línea de vista sea perpendicular al área de interés, paralela o perpendicular a algún eje de simetría del objeto. Otra condición que es muy importante en un sistema de visión es la iluminación, ya que es determinante para el funcionamiento del sistema, ya que en caso de no tomar esta condición en cuenta, pueden aparecer efectos no deseados tales como sombras y reflejos y el funcionamiento del sistema podría variar de tal forma que no se logren obtener los resultados requeridos. En cualquier caso, la inspección requiere de una comparación entre un modelo ideal y el objeto que se está inspeccionando [Martinsanz08].

2.2. Aplicación de los robots en la industria

El objetivo de la robótica es realizar diferentes tareas y trabajos de una manera más sencilla y eficiente. Las funciones de la visión computacional en la robótica es dar seguridad, verificando el estado del sistema en general. Otro objetivo de la visión computacional en la robótica industrial es dar mayor flexibilidad, es decir, permitir la adaptación del sistema cuando haya cambios y aumentar la tolerancia a errores de posicionamiento pero sin perder la eficiencia del trabajo. Un objetivo que se busca en la industria es abaratar costos, con esto se sustituye la mano de obra pero a la vez también se disminuyen los riesgos para las personas.

Los robots móviles tienen una amplia aplicación dentro de la industria, entre las que se encuentran el transporte de materiales peligrosos, robots desactivadores de explosivos, exploración de áreas que el hombre no puede explorar entre los cuales destacan algunos robots desarrollados por la NASA como lo son el Spirit y Oportunity, ver figura 2.1 [limoncellodigital] y figura 2.2 [roboticspot].

Figura 2.1: Spirit

Figura 2.2: Opportunity

Una aplicación más específica de los robots seguidores de línea es en una empresa maquiladora, el robot se puede usar para transportar materiales siguiendo una línea, usualmente la línea que sigue el robot es de color negro.

Las aplicaciones de transferencia de material se definen como operaciones en las cuales el objetivo principal es mover una pieza de una posición a otra. Las aplicaciones de transferencia de material normalmente necesitan un robot poco sofisticado.

En los robots se han encontrado un gran número de aplicaciones dentro de los laboratorios, ya que llevan acabo tareas repetitivas con un alto porcentaje de efectividad como la colocación de tubos de pruebas dentro de los instrumentos de medición. Los robots se han utilizado para realizar procedimientos manuales de manera automatizada. Un sistema típico de preparación de muestras consiste de un robot y una estación de laboratorio, la cual contiene balanzas, dispensarios, centrifugados, racks de tubos de pruebas, entre otros. Las muestras son movidas desde la estación de laboratorios por el robot bajo el control de procedimientos de un programa. Los fabricantes de estos sistemas suelen mencionar tener tres ventajas sobre la operación manual, como son:

- Incremento de la productividad.
- Mejoramiento del control de calidad.
- Reducción de la exposición del ser humano a sustancias tóxicas.

Las aplicaciones de estos sistemas en los laboratorios incluyen la medición del pH, viscosidad, preparación de plasma humano para muestras para ser examinadas, calor, flujo, peso y disolución de muestras para presentaciones espectromáticas, como se muestra en la figura 2.3[Mateos].

Existe una gran clase de aplicaciones en donde el robot realiza trabajo sobre pieza, este trabajo la mayoría de veces necesita que la pieza final del robot que efectúa el trabajo sea una herramienta y no una pinza, por lo tanto, la utilización de una herramienta para efectuar el trabajo es una característica diferente en este grupo de aplicaciones. El tipo de herramienta depende de la operación de procesamiento que se realiza.

- Soldadura por puntos.
  - La soldadura por puntos es un proceso en el que dos pieza de metal se soldan en puntos localizados al hacer pasar una gran corriente eléctrica a través de las piezas donde se efectúa la soldadura.

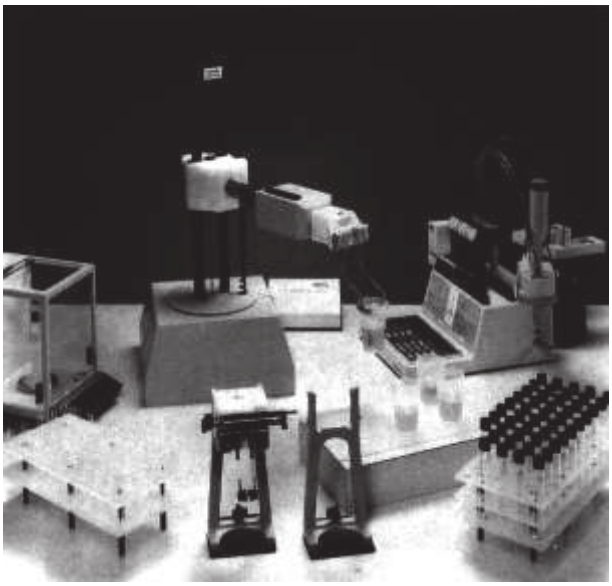


Figura 2.3: Laboratorio de forma automatizada

- Soldadura por arco continua.
  - La soldadura por arco es un proceso de soldadura continua en oposición a la soldadura por punto que podría llamarse un proceso discontinuo. La soldadura de arco continua se utiliza para realizar uniones largas o grandes en las cuales se necesita un cierre hermético entre las dos piezas de metal a unir. El proceso utiliza un electrodo en forma de barra o alambre de metal para suministrar una corriente eléctrica alta de 100 a 300 Amperes.

Además de la soldadura por punto y la soldadura por arco, existe una serie de aplicaciones de robot que utilizan alguna forma de herramienta especializada como efector final. Entre estas operaciones se encuentran:

- Taladro, acanalado y otras aplicaciones de mecanizado.
- Rectificado, pulido, cepillado, etc.
- Remachado.
- Corte por láser.

Los robots tienen una aplicación también en la agricultura, aunque para muchas personas se veía en esto una idea muy lejana o solamente ficción, la realidad es muy diferente por lo menos para el instituto de investigación Australiano, el cual ha invertido una gran cantidad de dinero y tiempo en el desarrollo de este tipo de robots, entre los cuales se encuentra una máquina que esquila ovejas, la trayectoria del cortador sobre el cuerpo de las ovejas se planea con un modelo geométrico de la oveja. Para compensar el tamaño de la oveja real y el modelo, se tiene un conjunto de sensores que registran la información de la respiración del animal como de su mismo tamaño, ésta es mandada a una computadora que realiza la compensaciones necesarias y modifica la trayectoria del cortador en tiempo real. Debido a que existe una escasez de trabajadores en los obradores, se encuentra en desarrollo otro proyecto, en el cual el objetivo es realizar un sistema automatizado de un obrador, el prototipo requiere un alto nivel de coordinación entre una cámara de video y el efector final que realiza en menos de treinta segundos, ocho cortes al cuerpo del cerdo [APL].

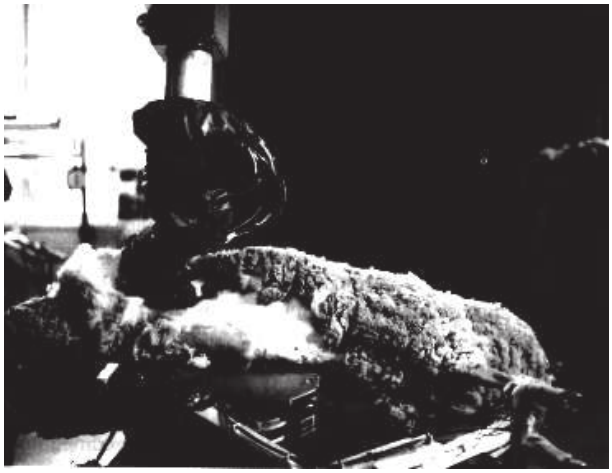


Figura 2.4: Sistema esquilador de ovejas.

Capítulo 3

Visión Computacional y la Librería OpenCV

3.1. ¿Qué es visión por computadora?

Existen diferentes definiciones sobre lo que es visión, una de las que se considera que se acerca más a lo que es visión es la siguiente: "Visión es un proceso que produce a partir de imágenes del mundo exterior una descripción que es útil para el observador y que no tiene información irrelevante."[Marr10].

Para entender un poco más lo que es visión, se mostrará la definición que se tiene sobre visión según Aristóteles, que nos dice que "Visión es saber que hay y donde mediante la vista".

Basándose en la descripción de [Marr10] sobre lo que es visión, se puede decir que visión por computadora es el proceso de captura, análisis, identificación de patrones útiles de una imagen digital para interpretarla y ejecutar alguna acción. Los sistemas de visión por computadora realizan tareas de inspección con un alto nivel de flexibilidad y repetición, nunca se cansan o distraen y pueden trabajar en ambientes donde un inspector humano no podría realizar la inspección visual o le costaría mucho trabajo realizarla.

Los sistemas de visión artificial ven al objeto por medio de un dispositivo de captura de imágenes, puede ser una cámara, un kinect, por mencionar algunos, los cuales

interpretan y procesan la imagen a través de una aplicación en un sistema computarizado para en seguida realizar alguna acción si ésto es requerido en el sistema. Las aplicaciones en visión por computadora se encuentran principalmente en tareas de inspección y ensamblaje, ya que estos trabajos son considerados repetitivos. En los seres humanos, la eficiencia en trabajos repetitivos se encuentra entre un 70 % y 85 % [Ortiz06].

3.2. Ventajas de visión por computadora

La visión por computadora tiene ventajas en tareas que se realizan a velocidades muy altas y en las tareas que requieren estar repitiéndose constantemente, por ejemplo en la inspección de tareas o procesos de ensamblaje y que se trabaje sin interrupciones. Existen tareas o procesos donde se requiere inspección visual. Los sistemas de visión tienen una clara ventaja sobre la visión humana, ya que los margenes de error son altamente reducidos. Se evitan algunos factores como el cansancio ya sea visual, muscular o distracciones, además de que disminuye el personal para realizar las inspecciones y/o revisiones por lo que se tiene un costo menor y más preciso a la hora de realizar una inspección con un sistema de visión artificial a comparación con los inspectores humanos. Otra ventaja de un sistema de visión por computadora o visión artificial es que se pueden realizar tareas con un horario amplio, es decir, se pueden realizar tareas durante largos periodos de tiempo sin que esto requiera un costo muy elevado debido al pago de horas extras al personal.

3.3. Desventajas de la visión por computadora

Una de las desventajas de la visión por computadora son las limitaciones que existen en la tecnología actual, por ejemplo, en comparación con la visión de nuestros ojos y el cerebro, los sistemas de visión computacional son muy básicos, además de que en un humano la velocidad de interpretación es muy alta en comparación con un sistema de visión artificial.

Otro problema que se presenta en un sistema de visión artificial son los problemas de iluminación, algunos sistemas deben contar con cierta iluminación para poder ejecutar

de manera correcta las tareas que debe realizar.

3.4. Componentes de un sistema de visión por computadora

Los principales componentes de un sistema de visión por computadora son: Una cámara de video o algún objeto similar para captura de imágenes y el software necesario para desarrollar la aplicación. La función de las cámaras de vídeo es capturar la imagen proyectada para mandarla a un sistema electrónico para que pueda ser interpretada, almacenada y/o simplemente visualizada. El sistema electrónico puede ser una computadora para visualizar, almacenar y procesar la imagen como lo es en el caso de este proyecto. El software es un elemento muy importante en una aplicación de visión por computadora ya que es el encargado de analizar, procesar e identificar las características de las imágenes capturadas.

3.5. Funcionamiento de un sistema de visión por computadora

El funcionamiento de un sistema por computadora consiste en adquirir una imagen por medio de un dispositivo, el cual puede ser una cámara web, un kinect, etc. una vez que se obtuvo la imagen, se realiza un preprocesamiento de la misma para resaltar las características necesarias. La siguiente etapa se llama segmentación o extracción de características. Esta etapa del sistema, se buscan los datos que cumplan con ciertas características para posteriormente procesar las imágenes procesadas y realizar las operaciones que sean necesarias para llegar al resultado deseado.

3.5.1. Adquisición de la imagen

En esta etapa se obtiene la imagen con la que se va a trabajar y la cual se puede obtener mediante diferentes dispositivos, tal como son una cámara o un kinect, en nuestro caso será una cámara web.

3.5.2. Preprocesamiento

El objetivo de esta etapa es la de mejorar la imagen obtenida resaltando determinadas características o eliminando los elementos que no se consideren necesarios.

3.5.3. Extracción de características o segmentación

Se busca dentro de la imagen datos que cumplan con ciertas características para resaltarlas y extraerlas para que sean procesadas más adelante.

3.5.4. Procesamiento de imágenes

El objetivo principal del procesamiento de imágenes es la realización de operaciones y transformaciones de una imagen digital para extraer el contorno, color, posición de los objetos y los niveles de iluminación en las imágenes digitales para lograr interpretarlas eliminando datos que no se necesitan lo cual se puede realizar mediante software para el análisis de imágenes. La iluminación es una parte muy importante dentro de los sistemas de visión computacional. En la actualidad, las cámaras son de menor calidad que la visión humana, por lo tanto, las condiciones de iluminación deben ser lo más óptimas para que una cámara pueda capturar una imagen que se pueda procesar e interpretar sin necesidad de iluminación extra. En la figura 3.1 se muestra el diagrama de un sistema de visión por computadora además de sus componentes.

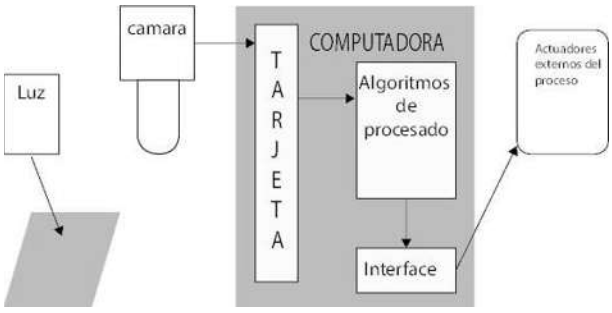


Figura 3.1: Diagrama de un sistema de visión

3.6. Librería OpenCV

OpenCV es una biblioteca de código abierto (también conocida como open source) para aplicaciones de procesamiento de imágenes y visión por computadora. Puede ser utilizada tanto en aplicaciones académicas y comerciales bajo la licencia BSD la cual permite que sea utilizada libremente, distribuida y adaptada según sean las necesidades que se tengan [Bradski08].

OpenCV nos ofrece una plataforma de código abierto para el procesamiento y manipulación de imágenes y video. En la figura 3.2 se muestra el logo de la biblioteca, dicha biblioteca existe en versiones para distintos lenguajes de programación y plataformas tales como C/C++, Java, Android, Python entre algunos otros. Cabe resaltar que debido a su condición de multilenguaje, se trata también de un entorno multiplataforma, es decir, que puede ser utilizada en cualquier sistema operativo que soporte los lenguajes mencionados anteriormente.

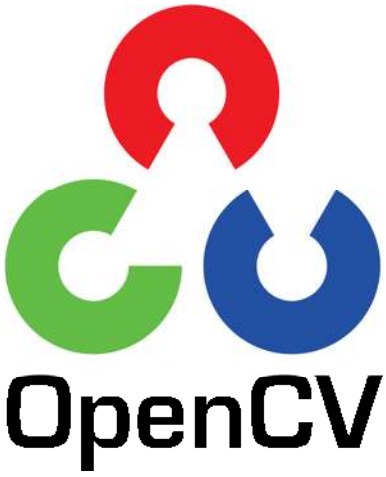


Figura 3.2: Logotipo de OpenCV

Algunas de las áreas de aplicación de la librería OpenCV [Corporation15] son:

- Herramientas de manipulación de características en 2 y 3 dimensiones.
- Estimación del movimiento en 3D de una cámara dentro de un ambiente.
- Sistema de reconocimiento facial.

- Reconocimiento de gestos.
- Interacción humano-computadora.
- Robótica Móvil.
- Identificación de objetos.
- Segmentación y reconocimiento.
- Visión estereo: percepción de profundidad de 2 cámaras.
- Estructura de movimiento.
- Seguimiento de objetos.
- Realidad aumentada.

Su descarga es gratuita desde el sitio web [opencv], en la cual además se pueden encontrar la documentación completa, tutoriales y foros de discusión relacionadas con la plataforma de la comunidad de programadores.

Capítulo 4

Hardware del robot

En este capítulo se describen las características de los componentes de hardware que constituyen el robot móvil autónomo.

4.1. Raspberry Pi

Raspberry Pi es una computadora de placa reducida o placa única de bajo costo (SBC por siglas en inglés de Single Board Computer), desarrollada en el Reino Unido por la fundación Raspberry Pi, con el objetivo de estimular la enseñanza de las ciencias de la computación en las escuelas. Es una computadora muy parecida a las que nos son familiares hoy en día, pero al utilizar un tipo de procesador diferente, no es posible instalar Microsoft Windows en ella. Sin embargo se puede instalar alguna de las diferentes versiones del sistema operativo Linux que se parecen y manejan como Windows. Es posible navegar en internet, enviar correos o escribir en un procesador de textos y además de realizar prácticas de programación [Upton12].

Las características de la microcomputadora para la versión B son:

- 512 en RAM.
- Unidad Central de Procesamiento: ARM1176JZ-F a 700 Mhz.
- Procesador gráfico GPU: Dual Core VideoCore IV Multimedia Co-Processor.

Figura 4.1: Raspberry pi

- Conector MIPI CSI que permite instalar un módulo de cámara desarrollado para la microcomputadora por la fundación.
- Puerto GPIO de 40.
- 2 puertos USB.
- Slot para tarjeta de datos SD.
- Energía: 750mA hasta 1.2A a 5V.
- Salida de video HDMI.

4.2. Sensor (Cámara)

En este caso se ha empleado la cámara para la microcomputadora Raspberry, la cual fue diseñada y fabricada por la fundación Raspberry Pi en el reino unido. Dicha cámara se conecta directamente a uno de los conectores serial CSI (Camera Serial Interface por sus siglas en inglés), que contiene la microcomputadora, el módulo se conecta a la Raspberry Pi a través de un cable plano de 15 pines a la interfaz serial (CSI), que fue diseñado especialmente para la conexión a las cámaras. Esta cámara es capaz de obtener imágenes a una resolución de hasta 5 mega pixeles (MP), 2594 pixeles (px) de ancho por 1944 de alto, además de capturar vídeo a 30 cuadros por segundo con una resolución de 1080p. La placa en sí es muy pequeña, alrededor de 25mm x 20mm y 9mm, y un peso de poco más de 3g,

ver la figura 4.2, por lo que es perfecta para dispositivos móviles u otras aplicaciones donde el tamaño y el peso son importantes. La cámara es compatible con la última versión del sistema operativo Raspbian, el sistema operativo preferido por los usuarios de la Raspberry Pi. A continuación se muestran algunas características de la cámara:

- Compatible con el modelo A y el modelo B raspberry Pi.
- Módulo de cámara 5MP omnivisión 5647.
- Resolución de imagen: 2592 x 1944.
- Video: soporta 1080p a 30 fps, 720p a 60fps y 640x480p 60/90 de grabación.
- 15 pin MIPI para la interfaz serial (se conecta directamente a la Raspberry Pi).
- tamaño: 20x25x9 mm.
- 3g peso.

Figura 4.2: Camára Pi diseñada para la conexión en un puerto SCI de la Raspberry Pi.

4.3. Puente H

El puente H o puente en H es un circuito electrónico que permite a un motor eléctrico de corriente directa (DC) girar en ambos sentidos, es decir, avanzar y retroceder. Los puentes H ya vienen hechos en algunos circuitos integrados, pero también se pueden construir a partir de componentes discretos.

Un puente H se construye con cuatro interruptores (mecánicos o mediante transistores), ver la figura 4.3. Cuando los interruptores S1 y S4 están cerrados (S2 y S3 permanecerán abiertos) se aplica una tensión positiva en el motor, haciéndolo girar en un sentido. Abriendo los interruptores S1 y S4 (se produce el cierre de S2 y S3), el voltaje se invierte, permitiendo el giro en sentido inverso del motor tal como lo muestra la figura 4.3 [Medina].

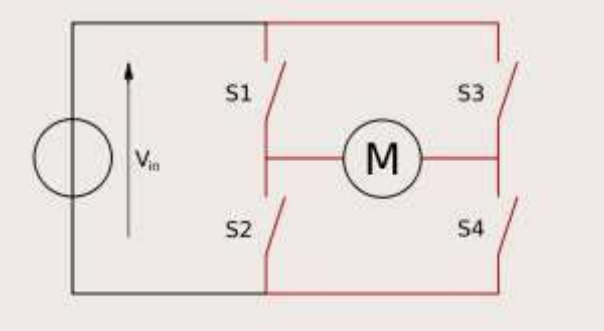


Figura 4.3: Puente H representado mediante interruptores.

Para lograr el control de velocidad y dirección de giro de los motores, se hace necesario el uso del ya mencionado puente H. En este proyecto se utilizó el encapsulado tipo DIP SN754410NE [Instruments98], ver la figura 4.4. Las características técnicas de este dispositivo son:

- Alimentación digital: 4.5V a 5.5V.
- Alimentación de carga: 36V máximo.
- Corriente de consumo: 70mA.
- Corriente de carga: 1.1 A máxima continuo.
- Frecuencia de operación: 0 a 1 MHz.

En la tabla 4.1 se especifican los pines del dispositivo SN754410NE mostrado en la figura 4.4.

Las señales de control de giro y velocidad utilizadas para el control de los motores son los pines 2 y 7 para el motor M1, los pines 10 y 15 son utilizados para el motor M2. Los

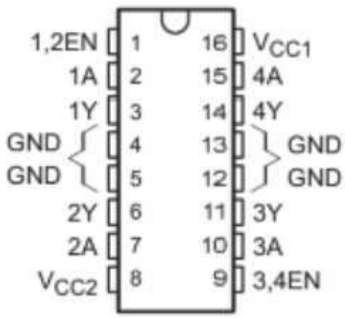


Figura 4.4: Disposición de los pines del puente H SN754410NE.

Pin	**Descripción**
1,2EN	Pin que habilita el canal 1 y 2
2 1A	Entrada digital del canal 1
3 1Y	Salida de carga del canal 1
4 GND	Alimentación Negativa
5 GND	Alimentación Negativa
6 2Y	Salida de carga del canal 2
7 2A	Entrada digital del canal 2
8 VCC2	Alimentacion de carga (0V - 36V)
9 3,4EN	Pin que habilita el canal 3 y 4
10 3A	Entrada digital del canal 3
11 3Y	Salida de carga del canal 3
12 GND	Alimentación negativa
13 GND	Alimentación negativa
14 4Y	Salida de carga del canal 4
15 4A	Entrada digital del canal 4
16 VCC1	Alimentación digital (5V)

Tabla 4.1: Pines del SN754410NE.

pines de habilitación son conectados directamente a una señal digital en alto (Vcc 5 Volts) como lo muestra en la figura 4.5, la cual representa el diagrama de conexión entre el circuito integrado SN754410NE y la tarjeta Arduino Nano, observar que en esta configuración se utilizan 4 señales PWM de la tarjeta Arduino Nano para realizar el control de giro y velocidad.

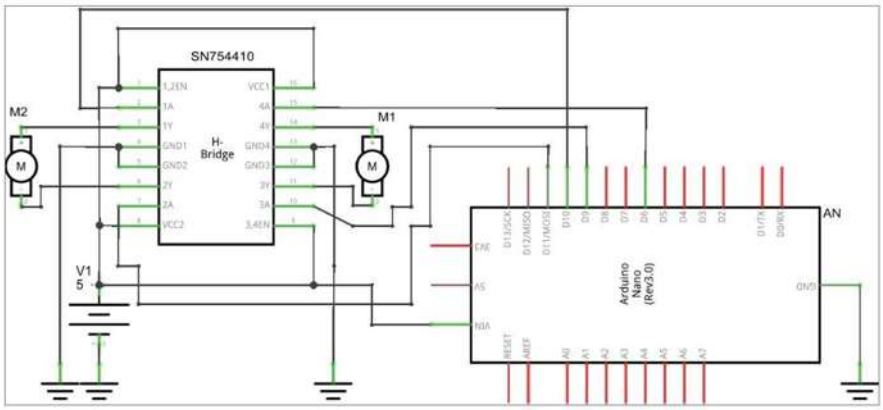


Figura 4.5: Diagrama de conexión del circuito integrado SN754410NE (puente H) y la tarjeta Arduino Nano.

4.4. Motores

El robot se mueve utilizando motores. Dependiendo del tamaño, el peso, la presición del motor, entre otros factores , éstos pueden ser de varias clases: motores de corriente continua, motores de paso o servomotores y motorreductores. En este caso, se utilizan 2 motorreductores de corriente directa. Las características de estos motorreductores se tienen en la tabla 4.2.

Voltaje CD (Volts)	**Corriente(miliamperes,mA)**	**Velocidad (RPM)**	**Torque (gf.cm)**
6	250	125	800

Tabla 4.2: Características de los motorreductores.

En la figura 4.6 se muestra la imagen de los motores utilizados para este proyecto.

4.5. Ruedas

Las ruedas del robot son movidas por los motores. Normalmente se usan ruedas de materiales anti-deslizantes para evitar fallas de tracción. Su tamaño es otro factor que se debe tener en cuenta a la hora de armar el robot, el diámetro de las llantas disponibles para el proyecto es de 6cm. En este caso, se utilizaran 2 ruedas, las cuales serán giradas por un motor cada una, además de una rueda loca que permite direccionar libremente al robot,

Figura 4.6: Motorreductores.

así que ello facilita los giros del robot. En la figura 4.7 se observan las llantas del robot que se usaron para el armado del mismo.

Figura 4.7: Ruedas del robot.

4.6. Tarjeta de control

La toma de decisiones y el control de los motores están generalmente a cargo de un microcontrolador. La tarjeta de control contiene dicho elemento junto con otros componentes básicos que requiere el microcontrolador para funcionar, en este caso, se utilizó una tarjeta Arduino Nano.

En la actualidad existe una variedad de tarjetas de desarrollo Arduino, observando las especificaciones, esta tarjeta cumple con las necesidades requeridas de nuestro sistema, considerando la parte de disponibilidad, economía y tamaño de la tarjeta, se utilizó la tarjeta Arduino Nano. Arduino Nano es una tarjeta de desarrollo basada en el microcontrolador Atmel ATmega168 o ATmega328, dependiendo de la versión disponible.

La función de la tarjeta en el robot móvil autónomo es realizar los cálculos del control PID y emitir las señales adecuadas para los motores, es decir, realiza el control de velocidad de los motores. A través de esta tarjeta de desarrollo es posible la conexión de dispositivos que tienen la capacidad de percibir el ambiente (sensores), los cuales pueden ser analógicos o digitales. En nuestro caso, el dato sensorial lo proporcionara la microcomputadora que tiene conectado el sensor principal de nuestro sistema (una cámara) realizando la comunicación entre los dispositivos por el puerto serie virtual que contiene la tarjeta Arduino Nano y el puerto USB de la microcomputadora.

Las especificaciones técnicas principales de la tarjeta se encuentran en la tabla 4.3:

Elemento	**Descripción**
Microcontrolador	Atmel ATmega168 o ATmega328
Voltaje de Operación (nivel lógico)	5 Volts
Voltaje de entrada (recomendado)	7,12 Volts
Voltaje de entrada (limites)	6-20 Volts
Pines Digitales Entrada/Salida (E/S)	14(de los cuales 6 proporcionan señales de salida PWM)
Pines de Entradas Analógicas	8
Corriente Directa por Pin de E/S	40mA
Memoria Flash	16 KB(ATmega168) o 32KB(ATmega328) de los cuales 2KB son utilizados por el sistema de arranque de la tarjeta (bootloader)
SRAM	1KB (ATmega168) o 2KB (ATmega328)
EEPROM	512 bytes(ATmega168) o 1KB (ATmega328)
Velocidad de Reloj	16MHz
Dimensiones	0.73”x 1.70”

Tabla 4.3: Tabla de especificaciones de Arduino.

En la figura 4.8 se muestra el Arduino Nano utilizado en el robot móvil.

Para mayor detalle de las especificaciones técnicas y funcionalidad de la tarjeta dirigirse al manual de usuario del Arduino Nano [Arduino08].

4.7. Baterías

La información de consumo de corriente que se tiene de las especificaciones de los elementos de hardware que constituyen nuestro sistema móvil se muestran en la tabla 4.4.

Figura 4.8: Arduino Nano.

Dispositivo	**Especificación de Consumo de Corriente(mA)**
Rasspberry Pi	800
Arduino nano	200(max 500 mA utilizando todos los pines de salida)
2 motorreductores	500

Tabla 4.4: Especificaciones de las baterías del robot.

Para la alimentación de la Raspberry Pi se ha utilizado una batería Lipo de 2000 mA, recomendada para alimentar la tajeta Arduino por medio del puerto USB. En el presente trabajo se utilizó una segunda fuente de alimentación (batería) de 1000 mA para alimentar los motores y la tarjeta Arduino.

La figura 4.9 nos muestra el tipo de bateria que se utilizó en el presente trabajo, cabe mencionar que la duración de trabajo continuo del robot con estas baterías en promedio es por arriba de una hora.

Figura 4.9: Batería lipo a 11.4 volts del robot.

4.8. Fuente reguladora de voltaje

Para el suministro de energía (5 volts) de la SBC (Raspbery Pi) fue necesario implementar una fuente regulada de voltaje que nos proporcionará más de 1 Ampere, debido a que el sistema de protección tanto de temperatura como de corriente de la fuente utilizada inicialmente (LM7805CT) era activado después de unos minutos de operación [Instruments03]. En el mercado existen varias alternativas en el caso de necesitar corrientes superiores a 1A, pueden utilizarse los reguladores de la serie 78HXX, LM3XX, en cápsula TO-3. Otra alternativa para obtener una fuente regulada a 5v mayor a 2A, es implementar un arreglo en paralelo de dos fuentes reguladas LM7805 como se ilustra en el diagrama esquemático de la figura 4.10 [Sebastian12], en donde también se muestra la disposición de los dispositivos en una vista tridimensional.

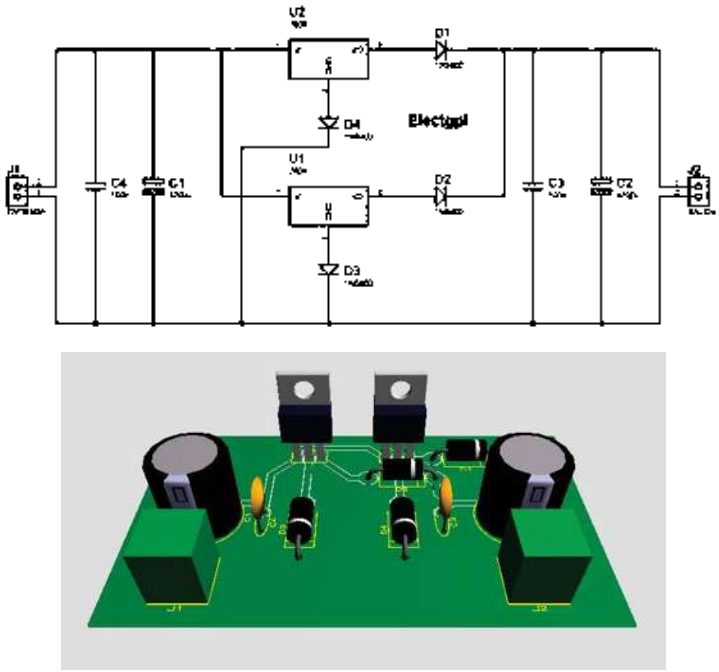


Figura 4.10: Fuente regulada de voltaje a 5 volts 2A.

4.9. Monitor del robot

Gracias al creciente desarrollo de pantallas táctiles, fue posible integrar una pantalla táctil de 3.5" al robot, esto permite interactuar con el robot y observar en tiempo real el vídeo que captura la cámara. Es importante señalar que por este dispositivo se verifica el comportamiento de los algoritmos en el procesamiento de imágenes para realizar la tarea del robot.

Se integró al robot una pantalla táctil de la marca Adafruit de 3.5", ver la figura 4.11, diseñada para ser utilizada con la SBC Raspberry Pi a través del puerto SPI en el puerto GPIO. Algunas especificaciones técnicas de la pantalla táctil [Himax12], se muestran en la tabla 4.5.

Figura 4.11: Display de 3.5"para la Raspberry Pi.

Tamaño	3.5 pulgadas en diagonal (LCD TFT display)
Resolución	320x480
Voltaje de alimentación	5V
Comunicación	SPI (Interfaz de Puerto Serial)
Dimensiones	56mm x 97mm x 2mm
Peso	52g

Tabla 4.5: Características de la pantalla táctil 3.5".

4.10. Bases del robot

Las partes en donde se soportan los elementos del hardware son las bases, ya que en ellas es en donde descansan todas las partes del robot como: la Raspberry Pi, las baterías, los motores y todos los componentes físicos que forman parte del robot. La construcción de las bases del robot se llevaron a cabo mediante un equipo CNC que recibe las instrucciones del lenguaje G. El lenguaje G o G-Code, es el nombre del lenguaje de programación que se utiliza para el control de máquinas de tipo CNC (Control numérico). Un programa escrito en este lenguaje es una lista secuencial de instrucciones que son ejecutadas por la máquina. Cada una de estas instrucciones representa un movimiento que debe realizar la máquina y el conjunto total de instrucciones representa todas las ordenes que se realizarán para el mecanizado de una pieza. Existen dos tipos de códigos, los códigos tipo G y los de tipo M. Los de tipo G representan funciones de movimiento de la máquina como son el avance, avance rápido, creación de arcos, pausas, etc., y los de tipo M representan funciones que no son de movimiento, como el cambio de herramienta, activación de la herramienta, activación del refrigerado, etc., al igual que los anteriores, estos tipos de códigos también son necesarios. Algunos de los códigos más utilizados son:

- G0. Movimiento rápido de la herramienta
- G1. Movimiento de avance lineal, hay que indicar la velocidad.
- G2. Interpolación circular, hay que indicar la velocidad y el radio
- G3. Interpolación circular, hay que indicar la velocidad y el centro.
- G04. Pausa
- G20 G21. Paso a pulgadas y a milímetros respectivamente.
- G28. Traslada automáticamente la herramienta a la posición de retorno cero predefinida.

Para más información sobre los códigos, consultar [Teruel04, GCO].

Utilizando acrílico como material para las placas, se realizó el soporte o placa de las baterías y los motorreductores para permitir al robot moverse sin que haya daño en sus componentes. El resultado de aplicar el código G son las placas del robot que se muestran en la figura 4.12, las cuales tienen una medida de largo de 180mm, y un ancho de 100mm además de dos perforaciones en la parte trasera y tres en la parte delantera. Se llevó a cabo el corte de una segunda placa para la parte superior del robot la cual es exactamente igual a la primera placa. En la figura 4.12 se muestra el resultado del corte de las placas finales y en la figura 4.13 se muestran las placas reales con el corte realizado.

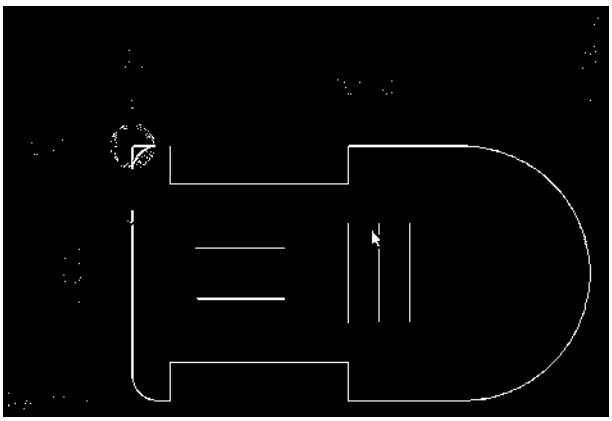


Figura 4.12: Dimensiones de la placa o base del robot móvil.

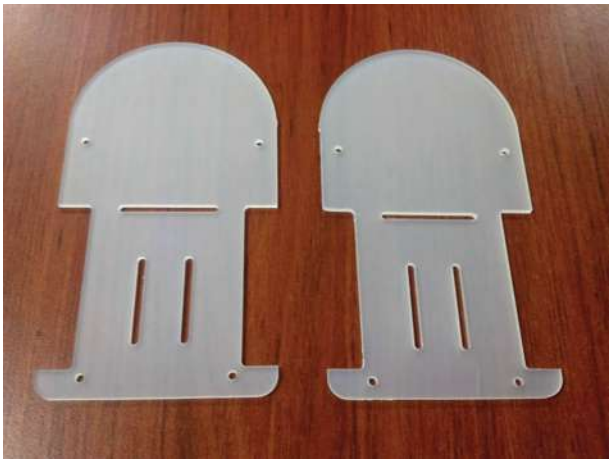


Figura 4.13: Placas del robot seguidor de línea.

En la figura 4.14 se muestra el robot una vez que se armó ya con todas sus piezas. Se pueden observar los componentes ya mencionados anteriormente una vez acomodados en el lugar correspondiente para el correcto funcionamiento del robot.

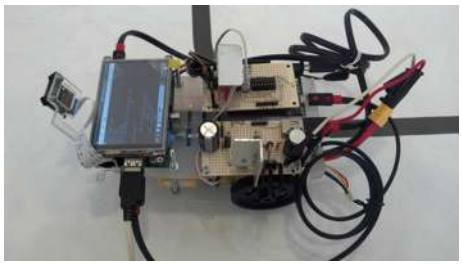


Figura 4.14: Figura del robot armado.

En la figura 4.15 se muestra una vista del robot armado desde una perspectiva frontal.

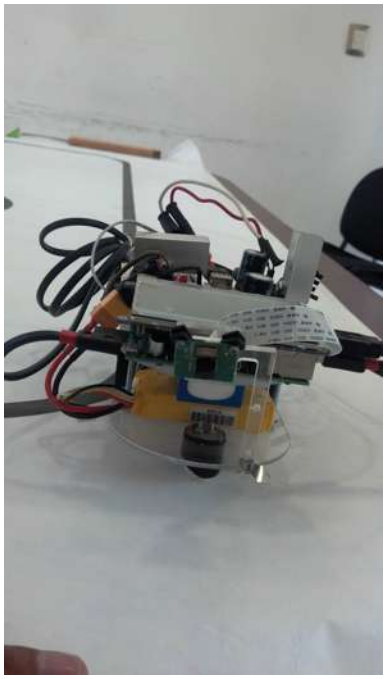


Figura 4.15: Vista frontal del robot.

Finalmente en la figura 4.16 muestra el diagrama de bloques de los componentes principales que conforman el sistema completo, en el cual se observa el elemento de visión

que en este caso corresponde a la cámara, la microcomputadora para el procesamiento de la información, la tarjeta de desarrollo para el sistema de control del robot y por último la interfaz con todos los elementos actuadores del robot, como es en este caso los motores.

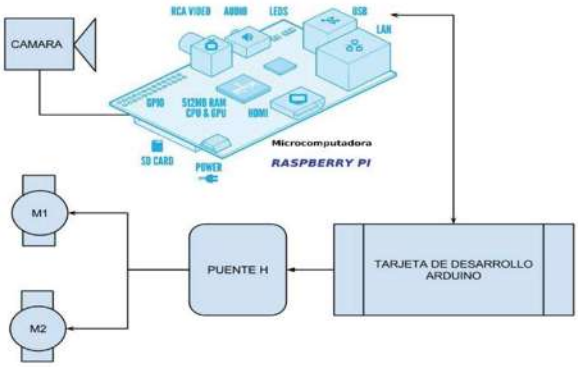


Figura 4.16: Diagrama general del sistema de visión con elemento de procesamiento, captura y bloques para el control del desplazamiento del robot.

Capítulo 5

Implementación del software para seguir una línea

Los robots seguidores de línea tienen un funcionamiento muy básico, el cual consiste en identificar el lugar donde se encuentra la línea y a partir de esa información retomar una dirección, en otras palabras, si el sensor detecta que la línea se encuentra a la derecha, el robot gira hacia la izquierda, de igual manera si el sensor detecta que la línea se encuentra a la izquierda, el robot realiza un giro hacia la derecha. Sin embargo, este proceso no es muy eficiente en carros de mayor velocidad ya que estaría cambiando constantemente de dirección y produciendo un movimiento en zigzag lo que provocaría que disminuya la velocidad de una manera notable. Para estos casos y evitar este problema, se le agrega un sistema de control el cual tiene la función de controlar en forma suave los desplazamientos que debe realizar para tomar la dirección correcta del carro, empleando cambios de velocidad en los motores.

5.1. Sistema de visión para seguidor de línea

Para la implementación y desarrollo del software para nuestro sistema de visión se consideró realizar el sistema en Python con la libreria de OpenCV, se eligió Python ya que se tienen algunas ventajas de usar este lenguaje como son:

- Multiplataforma.
- Permite programación multiparadigma.
- Ofrece un gran soporte y documentación que facilita en gran medida su desarrollo.
- Existe una implementación muy eficiente para la manipulación de la cámara diseñada para la SBC Raspberry Pi (`picamera` [Jones14]).

Con la unión de Python y OpenCV, se puede crear un código rápido y eficiente para el procesamiento de imágenes, controlar motores por medio de mandos seriales, y además permite diseñar mecanismos de control que permiten lograr un buen desempeño en los robots móviles.

Para la implementación de sistemas de visión, los problemas principales que se le presentan al desarrollador son:

- Eliminación de información no útil en la imagen (comúnmente conocido como ruido en la imagen).
- Segmentación de la imagen, se refiere a que una vez capturada la imagen identificar las regiones de interés que nos permitirán identificar características de los objetos a reconocer.
- Identificación del objeto de interés y establecimiento de parámetros para el sistema de control.

Todo lo anterior se debe implementar sin que la información importante captada por la cámara se pierda o se deteriore.

La implementación del sistema de visión para el prototipo seguidor de línea se presenta en el diagrama de bloques de la figura 5.1.

Las operaciones (bloques del diagrama en la figura 5.1), se ejecutan en un ciclo infinito mientras el sistema permanezca con energía. Se describe a continuación cada una de las operaciones de procesamiento que permiten la identificación de la línea a seguir y además calcular su posición en relación a la referencia indicada en la imagen.

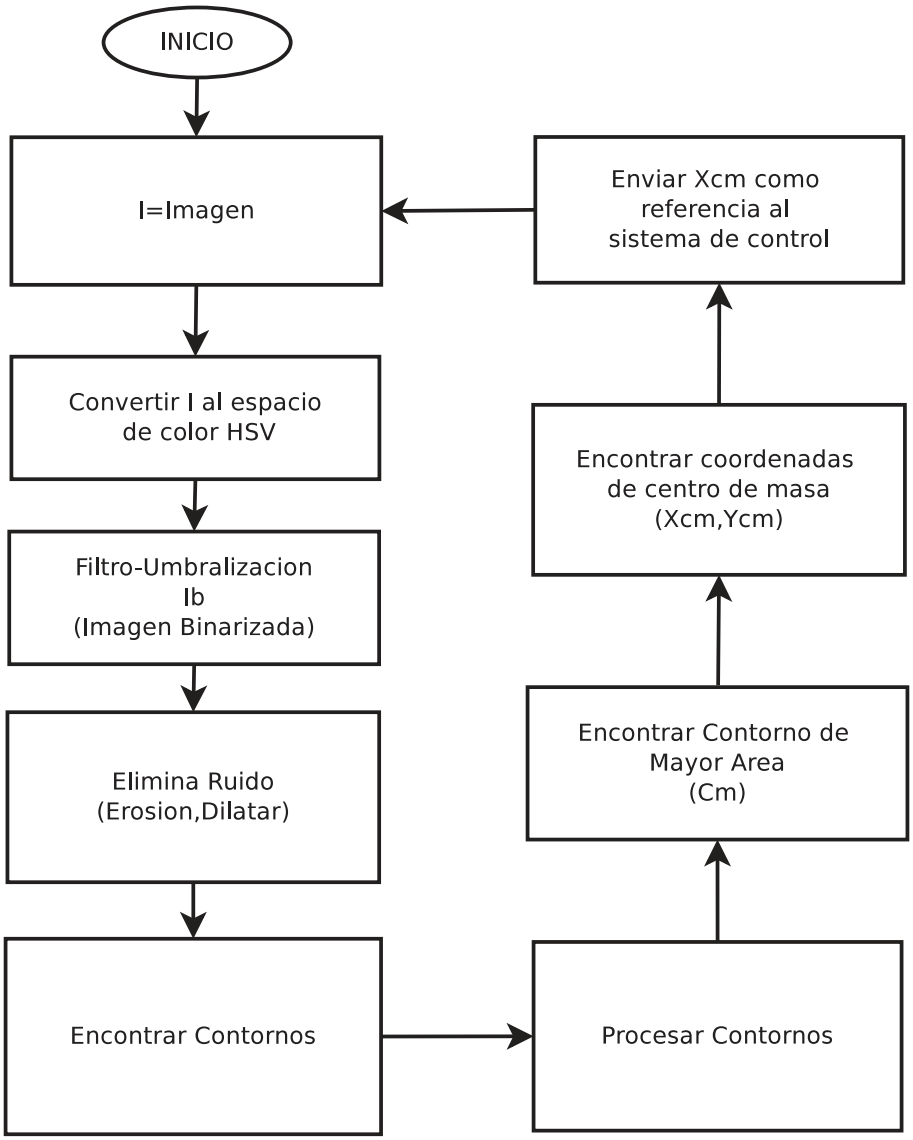


Figura 5.1: Diagrama de flujo del proceso en general que se realiza para la identificación del objeto (línea) en el sistema de Visión Computacional.

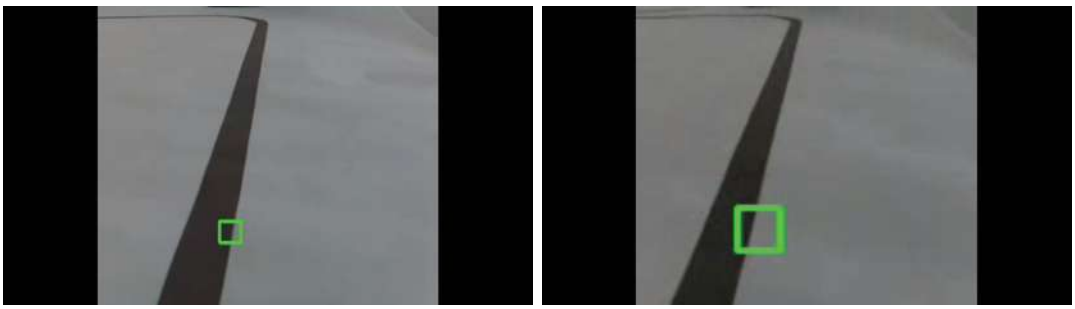
5.1.1. Adquisición de la imagen

Se captura el vídeo a diferentes resoluciones, tomando una de las imágenes que componen la secuencia de vídeo ($I = imagen$), comúnmente conocido como frame o marco. Gracias al driver especializado que se tiene para manipular la cámara conectada al puerto CSI de la Raspberry Pi, se logra procesar más de 5 marcos por segundo en está propuesta de programación, de acuerdo a las pruebas realizadas. La figura 5.2 nos muestra las imágenes

captadas por la cámara Pi a diferentes resoluciones. En base al ancho (número de columnas) y alto (número de renglones) de la imagen, se establece el centro de la imagen como coordenadas de origen que establecen la referencia de posición del objeto identificado, en nuestro caso la línea.

$$x_o = \frac{Ancho}{2}$$

$$y_o = \frac{Alto}{2}$$



(a) ***Resolución de 320x280px.*** **(b)** ***Resolución de 160x140px.***

Figura 5.2: Captura de Frames a resoluciones de 320x280px y 160x140px.

5.1.2. Espacio de color HSV

Dentro de las acciones que se realizan se encuentra la conversión de los Frames capturados al espacio de color HSV (siglas del inglés conformadas por Hue, Saturation an Value) donde Hue (Matiz) especifica el color, por ejemplo rojo, naranja, azul, etcétera [Laganiére11]. La saturación se refiere a la cantidad de blanco en el matiz. Un color completamente saturado no contiene blanco y aparece puro, un rojo saturado al 50 % resulta un rosa. El valor (también conocido como brillo) es el grado de luminosidad de un color. Una matiz con alta intensidad es brilloso, uno con poca intensidad es obscuro.

La transformación que se realiza al espacio de color HSV, nos ayuda a enfrentar los cambios de iluminación del ambiente y que afectan directamente al sistema, es necesario sintonizar el sistema a cambios de iluminación drásticos. La figura 5.3 nos muestra el resultado de la transformación de la imagen original al espacio de color HSV.

Se establecieron los rangos de valores HSV en el sistema que nos permiten detectar la línea negra, así el rango de H es de 0 hasta 115, S de 0 a 89 y de V de 0 a 80, que son los valores que le corresponden a la línea para que el sistema la detecte.

Figura 5.3: Transformación de la imagen original al espacio de color HSV.

5.1.3. Binarización de la imagen (Umbralización)

La umbralización de una imagen se refiere al proceso de filtrar los valores de los píxeles de acuerdo a un valor determinado, conocido como umbral, y establecer aquellos valores de píxeles mayores o menores al valor de umbral, en un valor binario (0 ó 1). Esta operación lo que finalmente realiza es proporcionar regiones de píxeles que son fuente de información útil para nuestro propósito, descartando parte de la información que se considera ruido en la imagen. La figura 5.4 es la imagen binarizada que observa el sistema de visión. Observar como elimina regiones que corresponden a información no útil para nuestro objetivo de detectar la línea negra. Además se utiliza solamente la mitad de la imagen (respecto del alto de la misma) para procesar la información a mayor velocidad y para acotar el área de identificación de la línea.

Figura 5.4: Imagen binarizada de la imagen original, se muestra solamente la mitad del alto de la imagen.

5.1.4. Eliminando Ruido por Erosión y Dilatación

Erosión y Dilatación son dos operadores morfológicos que se aplican a elementos estructurales en una imagen [Bradski08], tienen las siguientes aplicaciones:

- Remover ruido.
- Separar a elementos individuales las estructuras o unir elementos dispares en la imagen.
- Encontrar huecos en una imagen o intensas.

Con el operador de dilatación se logra adelgazar los elementos estructurales de la imagen cuando el fondo de ésta es blanco y los elementos de color negro. Mientras que el operador de Erosión bajo las mismas condiciones de la imagen respecto del color de fondo, logra hacer más burdos (engrosar) los trazos de los elementos en la imagen.

Al aplicar los operadores morfológicos de Erosión y dilatación a cada uno de los marcos en el sistema, se está eliminando ruido que se encuentra cerca de la línea a identificar, logrando con ello una mejor precisión en la detección de la línea. Las figuras 5.5 y 5.6 nos muestran el trabajo que realizan para el sistema.

5.1.5. Encontrar contornos

Los contornos pueden ser explicados como simples curvas uniendo todos los puntos continuos (a lo largo de un límite), los cuales tienen el mismo color o intensidad. Los

Figura 5.5: Eliminación de ruido por Dilatación.

Figura 5.6: Eliminación de ruido por Erosión.

contornos son herramientas útiles para el análisis de formas, detección e identificación de objetos [Bradski08].

En esta parte del proceso se persigue encontrar los contornos en la imagen binaria, para ello se emplea la función `FindContours()` que ofrece OpenCV. La técnica que se utiliza para la detección de contornos está descrita en el documento de [Suzuki85]. Ver la figura 5.7 en donde se observan los contornos encontrados en la imagen binaria a la que se le aplicó erosión y dilatación.

Procesar contornos

La estrategia a seguir para identificar la línea es buscar los contornos que cubren la mayor cantidad de área en la imagen. Sabemos entonces que los contornos potencialmente candidatos a ser el objeto a identificar son todos aquellos que cubren una área mayor en la imagen, por lo tanto se procesan los contornos para eliminar o filtrar los que no cumplen

(a) ***Línea detectada de forma binaria.*** **(b)** ***Contornos de la línea detectada.***

Figura 5.7: Imagen de la captura de contornos e imagen binarizada.

con un valor de área establecido.

Para ello se fija un valor de umbral (T_a) y todos aquellos contornos menores son descartados, esto nos permite descartar contornos muy pequeños y si hay algún ruido lo omita y solamente toma en cuenta los contornos mayores.

Para discriminar los contornos se hace uso de las características que en forma inherente existen en ellos. En este punto de nuestro proceso, obtenemos entonces el contorno de mayor área, el cual identifica el objeto que deseamos seguir con el robot móvil.

Centro de Masa del Contorno con Mayor Área (C_m)

Una vez que se detectó el contorno con el área mayor, el cual corresponde a la línea, procedemos a encontrar las coordenadas del centro de masa (X_{cm}, Y_{cm}) del área que se encontró, se hace la traslación de esta coordenada a las coordenadas de origen (x_o, y_o), obteniendo valores positivos o negativos de los valores de las coordenadas. Se marca el centro de masa con un recuadro de color como señal de la posición en donde se encuentra la referencia a seguir y también como indicativo del trabajo que se realiza por el procedimiento.

5.1.6. Envío de dato de referencia

Finalmente, Los valores negativos de x_o nos indican que la línea se encuentra en el lado izquierdo y que el robot debe girar a la derecha para alinearse al centro de la pantalla. En el caso de $x_o > 0$ (valor positivo), la línea se encuentra en el lado derecho del robot y se debe girar hacia la izquierda. Está información se envía por medio del puerto USB de la

SBC a la tarjeta Arduino que la lee como un puerto de comunicación serial.

5.1.7. Parámetros ajustables en el sistema

Debido a que el sistema de visión propuesto se encuentra sujeto a diferentes condiciones de iluminación, requiere del ajuste de varios parámetros. Los primeros parámetros que se ajustan del sistema son las variables del espacio de color: H (Matiz), S (Saturación), V (valor), en el caso de este trabajo se ajustaron para que detecte una línea de color negro trazada en el piso, en el cual se realizó un video de prueba con el que se estuvo trabajando para el desarrollo del sistema. La línea tiene aproximadamente 2cm de ancho la cual tiene un contraste con un piso de fondo como lo muestra la figura 5.8, los valores que se proponen son: H=0-115, S=0-89 y V=0-80.

Figura 5.8: Imagen de la línea en contraste con el piso.

Otros parámetros que requieren ajustarse son:

- Ancho y alto de la imagen: se consideraron valores de 320x280 y 160x140 de ancho por alto respectivamente.
- Umbral del área de los contornos para descartar lo de menor área.

En la figura 5.9 se muestra la imagen de lo que se observa en el video de prueba con el que se trabajó para realizar algunas pruebas del sistema ya con el cuadrado verde que sigue la línea.

Figura 5.9: Imagen de video de prueba.

En la figura 5.10 se muestra los contrastes de la línea y lo que observa son los colores ya con la aplicación de las transformaciones de la imagen y los filtros para que se pueda detectar el color de la línea. Obsérvese que el color de la línea recta que en este caso es negro, se logra resaltar de color blanco al aplicar los filtros ya mencionados.

Figura 5.10: Imagen de video de prueba con filtros

Como ya se mencionó, se trabajó en la parte inferior de la pantalla que captura el robot, ya que aunque se pierde visión, se gana en velocidad además de que se tiene menos cantidad de que haya ruido y el sistema lo detecte.

5.2. Sistema de control del robot móvil

Existen varios tipos de sistemas de control que son utilizados para el control de robots móviles, el más simple de estos sistemas es el control Proporcional, Integral y Derivativo (PID por sus siglas en inglés de Proportional, Integral and Derivative). El control PID se ha utilizado para controlar alrededor del 90 % de los procesos en la industria por lo que lo convierte en la técnica de control más común [Åström09]. Se basa en minimizar el error que se obtiene de la diferencia de los valores medidos entre la salida y un valor que se establece como punto de referencia, resultando en un ajuste que controlan los actuadores del sistema, repercutiendo en las entradas.

Dentro de la robótica móvil el control PID se utiliza en el problema de seguidores de línea, en el control de robots auto-balanceados en dos ruedas, además de los seguidores de objetos por mencionar algunas aplicaciones. Las partes que conforman el control PID tienen los siguientes propósitos:

- La parte proporcional tiene como propósito la eliminación del error que se obtiene de la medición actual con el valor de referencia.
- La parte integral es utilizada para promediar el error a lo largo del tiempo, tiene la información promedio del error en el pasado.
- La parte derivativa se utiliza para predecir el error futuro a través de la variación del error en el pasado.

El valor final de control "u"puede ser calculado al ir sumando los tres términos descritos, dados por la siguiente ecuación:

$$u = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

Los valores de las ganancias en cada uno de los términos pueden obtenerse por varios métodos. De forma experimental se puede establecer a cero las ganancias de los términos integral y derivativo y probar con el término proporcional hasta observar un resultado deseado. Agregar los siguientes términos uno a uno con valores pequeños hasta

observar la salida deseada, tal y como se realizó en este trabajo. La figura 5.11 nos muestra el diagrama del sistema de control PID para el robot [Moisés14].

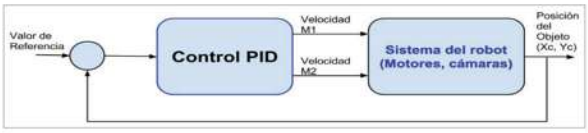


Figura 5.11: Diagrama de control del robot

Capítulo 6

Experimentación y resultados

El objetivo de la parte experimental es verificar que el robot móvil autónomo logre cumplir con la tarea de seguir la trayectoria que se dispone. Verificar para diferentes resoluciones del vídeo el comportamiento del sistema de visión y además diferentes voltajes de alimentación en los motores, lo cual produce un incremento en la velocidad.

6.1. Prueba No. 1: Seguimiento en línea recta

Una de las pruebas realizadas al prototipo es verificar que realmente puede seguir una línea recta. Para realizar esta prueba, se utilizó la impresión en papel Bond de una línea recta color negro la cual tiene como característica 2 cm de espesor. El largo de la línea que siguió el prototipo fue de 1.7m, ver la figura 6.1.

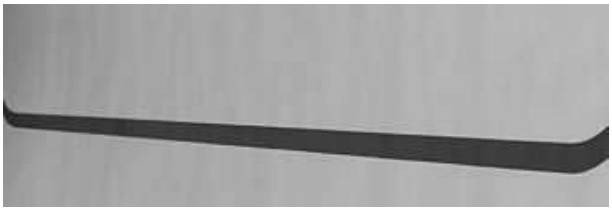


Figura 6.1: Pista de línea recta en papel bond de longitud 1.7 metros, espesor de 2cm., color de la línea negro con fondo blanco.

Durante esta prueba, en donde los motores estuvieron alimentados con 5 volts, el robot tardó 12 segundos en recorrer la línea observándose que la siguió en forma satisfactoria,

lo cual corresponde a una velocidad de

$$\frac{1.7m}{12s} = 0.1417\frac{m}{s}$$

ó $14.17\frac{cm}{seg}$. Durante el recorrido de la línea, se lograron capturar 60 Frames en total a una resolución de 320x280px (ancho por alto respectivamente) lo que corresponde a una velocidad de procesamiento de 5 frames por segundo(fps). Se procedió entonces a realizar la prueba del recorrido de la misma línea recta de 1.7 metros pero a una resolución más baja que corresponde a 160x140 píxeles, durante esta prueba, se logró recorrer la línea en un total de 9 segundos lo que corresponde a una velocidad de

$$\frac{1.7m}{8s} = 0.2125\frac{m}{s}$$

ó también $21.25\frac{cm}{seg}$, durante el recorrido de la línea, se lograron capturar 42 frames en total, obteniendo como resultado una velocidad de procesamiento de 4.6667 fps. En la tabla 6.1 se muestra en forma condensada los resultados descritos. Cabe mencionar que en esta prueba se aplicó un voltaje de 5 volts al dispositivo SN754410NE para alimentar los motores, siendo ésta la velocidad más baja de operación del robot.

Frames	**Tiempo (Segundos)**	**Velocidad de procesamiento (fps)**	**Resolución**	**Velocidad del Robot** ($\frac{cm}{s}$)
60	12	5	320x280	14.17
42	8	5.25	160x140	21.25

Tabla 6.1: Resultados del seguimiento de línea recta a diferentes resoluciones de video alimentando los motores a 5 volts.

En las figuras 6.2 y 6.3 se muestran imágenes de la visión que tiene el sistema a las dos diferentes resoluciones en que se realizaron las pruebas (320x280 px y 160x140 px respectivamente), en ellas se observa además el recuadro en color que identifica la línea. Debido a que se analiza únicamente la mitad inferior de la imagen visualizada, se elimina la línea negra de la parte superior en la imagen.

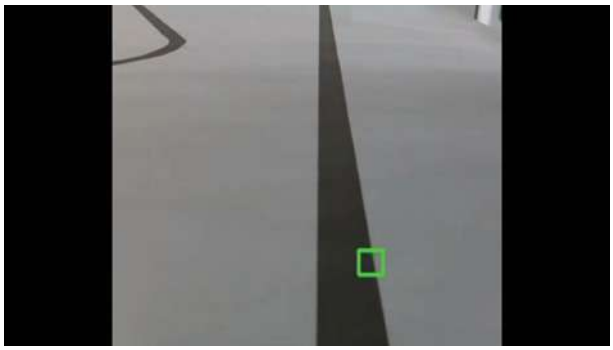


Figura 6.2: Vista del robot a una resolución de 320x280px.

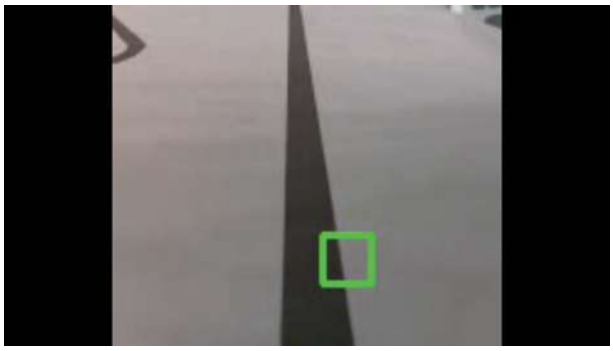


Figura 6.3: Visión del robot a una resolución de 160x140px.

6.2. Prueba No. 2: Trayectoria de línea curva sobre piso de concreto

En está prueba se utilizó una cinta de aislar color negro de ancho 1.8 cm. Para trazar la trayectoria de la línea, adherida a un piso firme de concreto se formó la trayectoria con algunos tramos curvos, como se muestra en la figura 6.4. Se mantuvo el voltaje de 5 volts de alimentación a los motores. La longitud de la trayectoria para esta prueba fue de 2.8 metros.

Durante esta segunda prueba realizada, se lograron capturar en promedio 4.315 fps, este dato nos indica que se tiene una capacidad baja de procesamiento por parte de la Raspberry Pi, en la tabla 6.2 se proporcionan los resultados de velocidad para esta prueba. Esta prueba se realizó a las diferentes resoluciones de 320x280px y 160x140px.

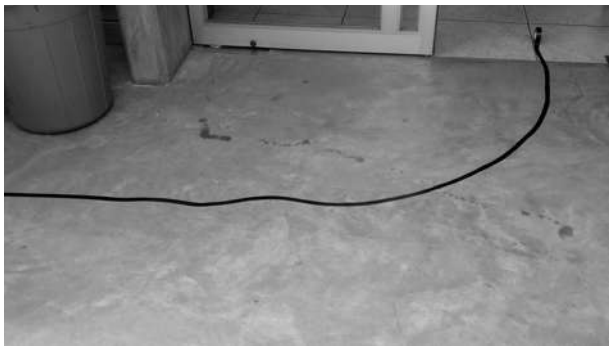


Figura 6.4: Pista trazada con cinta de aislar de ancho 1.8 cm., longitud de 2.8 m. sobre piso de concreto en un ambiente exterior.

Para la resolución de 320x280px se logró recorrer la pista en un total de 20 segundos lo cual corresponde a una velocidad de 0.14 m/s ó 14 cm/s, logrando capturar un total de 67 frames, obteniendo como resultado una velocidad de procesamiento de 3.35 fps. Para la resolución de 160x140px, se logró el recorrido de la línea en un tiempo total de 14 segundos, por lo que se logra una velocidad de 0.2m/s ó 20cm/s, además de que se lograron capturar 74 frames en ese tiempo, por lo que la velocidad de procesamiento para esta resolución es de 5.28 fps.

Frames	**Tiempo (Segundos)**	**Velocidad de procesamiento (fps)**	**Resolución**	**Velocidad del Robot** ($\frac{cm}{s}$)
67	20	3.35	320x280	0.14
74	14	5.28	160x140	0.2

Tabla 6.2: Resultados del seguimiento de línea en una pista no lineal sobre concreto a diferentes resoluciones de video a 5 volts.

Durante esta prueba, el robot siguió la trayectoria de la línea trazada de manera satisfactoria durante la prueba realizada a las dos diferentes resoluciones a las que se realizaron las pruebas que son a 320x280px y 160x140px.

6.3. Prueba No. 3: Seguimiento en línea recta con un voltaje de alimentación mayor

Debido a que el dispositivo SN754410NE nos permite alimentar con un voltaje de hasta 32 volts a los motores, se suministro el voltaje máximo de la batería al pin del puente H destinado para este propósito. El objetivo de este experimento es verificar el desempeño del sistema de visión, al incrementar la velocidad del robot móvil.

El aumentó del voltaje aplicado al dispositivo SN754410NE para alimentar los motores, paso de 5.0 volts a 11.4 volts. Para la longitud de la pista en línea recta de 1.7 m., se logró el recorrido satisfactoriamente en un tiempo de 6 segundos en la resolución de 160x140 píxeles, observando una velocidad de

$$\frac{1.7m}{6s} = 0.283\frac{m}{s}$$

ó $28.3\frac{cm}{s}$., se concluye que aún con el incremento de la velocidad se logró el objetivo de recorrer la trayectoria bajo estas circunstancias. En comparación con las primeras pruebas realizadas, el robot alcanza un incremento de velocidad del 19.3 % más. Durante esta prueba se lograron capturar un total de 40 frames en los 6 segundos que dura en recorrer la pista, obteniendo como resultado una velocidad de procesamiento de 6.667fps. En la tabla 6.3 se muestran los resultados obtenidos para esta prueba, en donde sólo se tuvo éxito en la resolución de prueba más baja.

Frames	**Tiempo (Segundos)**	**Velocidad de procesamiento (fps)**	**Resolución**	**Velocidad del Robot**$(\frac{cm}{s})$
40	6	6.667	160x140	28.3

Tabla 6.3: Resultados del seguimiento de línea en una pista lineal a una resolución de 160x140px de video.

En la figura 6.6 se muestra al robot en el momento en el que sigue la trayectoria de una línea recta.

Por otra parte, al momento de realizar la prueba para la resolución de 320x280 píxeles, por la velocidad del robot no se logró recorrer la trayectoria dispuesta en la tarea del robot, lo que se observó fue que no es posible capturar el número suficiente de frames,

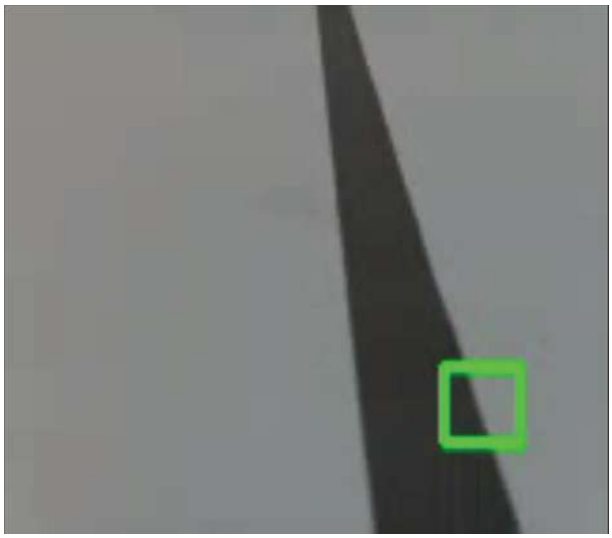


Figura 6.5: Visión del robot a una resolución de 160x140 px, en la prueba sobre una línea recta trazada sobre papel bond color blanco, alimentando los motores a 11.4 volts, espesor de la línea color negro de 2cm. y longitud de 1.7 m.

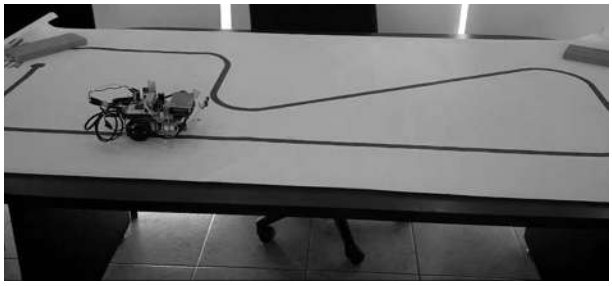


Figura 6.6: Robot realizando el seguimiento de la línea recta.

procesarlos y emitir el valor de referencia al sistema de control para una pronta respuesta, provocando que el robot salga de la línea.

6.4. Prueba No. 4: Seguimiento en una pista con curva a mayor voltaje

Esta prueba se realizó en una pista de una línea trazada sobre el pavimento con cinta de aislar, la cual tiene un ancho de 1.8cm de ancho y el trayecto dispuesto un largo de 3.3 m., la pista tiene además una curva como se puede observar en la figura 6.7. Con una

resolución de video de 160x140 píxeles el robot recorrió de manera satisfactoria la trayectoria en un tiempo de 16 segundos, con lo que se alcanzó una velocidad de

$$\frac{3.3}{16} = 0.203\frac{m}{s}$$

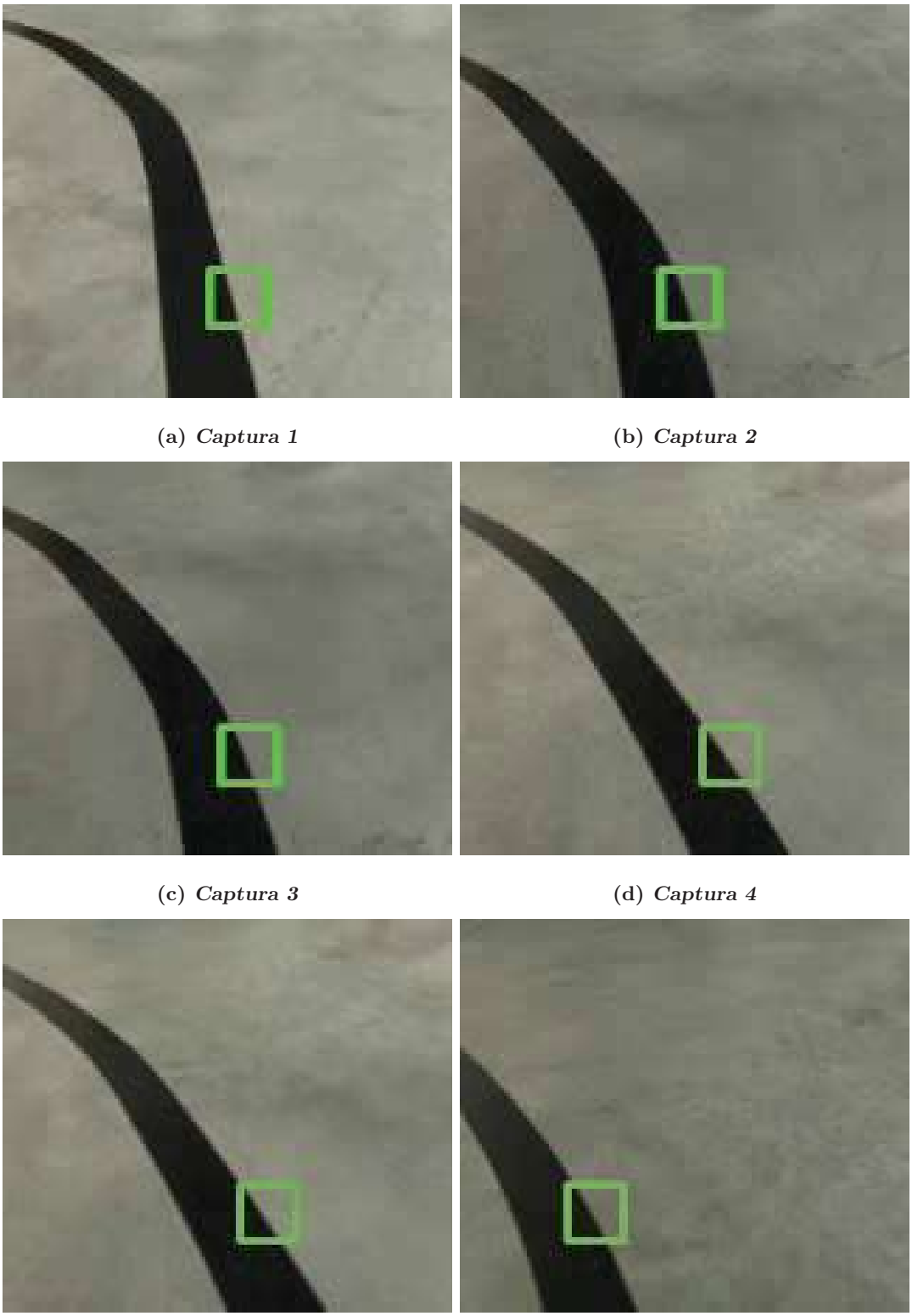
ó $20.3\frac{cm}{s}$. El incremento en la velocidad en esta prueba es muy notoria, además de que el robot recorrió la pista rápidamente y sin salirse de la línea trazada.

Figura 6.7: En la imagen se muestra la trayectoria de línea de color negro curva trazada en piso de concreto para el robot seguidor de línea.

Durante la prueba realizada, el robot logró capturar 84 Frames en un total de 16 segundos, logrando así una velocidad de captura de 5.25fps (Frames por segundo). En la figura 6.8 se observan 6 frames consecutivos obtenidos por el sistema de visión del robot durante el transcurso del recorrido de la trayectoria. La tabla 6.4 presenta los resultados obtenidos para esta prueba.

Frames	**Tiempo (Segundos)**	**Velocidad de procesamiento (FPS)**	**Resolución**	**Velocidad del Robot** $(\frac{cm}{s})$
84	16	5.25	160x140	20.6

Tabla 6.4: Resultados del seguimiento de línea en una pista con trazos curvos sobre concreto a una resolución de 160x140px de video.



(a) ***Captura 1*** (b) ***Captura 2***

(c) ***Captura 3*** (d) ***Captura 4***

(e) ***Captura 5*** (f) ***Captura 6***

Figura 6.8: Visión que obtiene el robot móvil autónomo durante la ejecución de la prueba No. 4.

Capítulo 7

Conclusiones y Trabajos Futuros

7.1. Conclusiones

En está sección se presentan las conclusiones del desarrollo de software e implementación del robot seguidor de línea con la Raspberry Pi y la librería de visión computacional OpenCV.

- Se implementó un prototipo de un robot móvil autónomo seguidor de línea que demuestra la factibilidad de uso de la computadora SBC Raspberry Pi como unidad de procesamiento en el robot.

- La velocidad de procesamiento promedio fue de 5 fps para identificar la línea y proporcionar su posición en el eje x, a una resolución baja fue posible realizar exitosamente la tarea del robot.

- Se muestra la aplicación de un sistema de visión y su factibilidad para el desarrollo de proyectos en el ámbito académico, además el prototipo puede ser utilizado como un recurso didáctico para estudiantes de la materia de visión computacional.

- Se cumplió con el objetivo de lograr integrar la Raspberry Pi, la tarjeta de desarrollo Arduino, y la implementación de la librería OpenCV para realizar la tarea del robot que es: seguir la línea mediante un sistema de visión computacional.

- Se implementó el sistema de control PID para que el robot realice un seguimiento de trayectoria suave; la sencillez del control es debido a que no se requiere una alta precisión en el sistema. Es posible utilizar esta implementación para cuestiones didácticas en las materias de control.

- Se enfatiza la sencillez con lo que es posible realizar el desarrollo de software para un sistema de visión implementado con la librería OpenCV y las librerías de optimización implementadas en Python. La poca cantidad de líneas de código en los apéndices C y D así lo demuestran.

7.2. Trabajos Futuros

1. Como trabajo futuro, este prototipo nos permitirá realizar demostraciones prácticas en materias como lo es visión computacional o sistemas de control.

2. Diseñar estrategias para enfrentar los cambios de iluminación en el ambiente y sintonizar de forma automática los parámetros que se requieren para la identificación de los objetos.

3. Con este prototipo de robot móvil, se podrá implementar por medio del Framework Django, un sistema web para manipular el robot, además de estar mostrando al mismo tiempo el video que capta la cámara en tiempo real por medio de videostreaming. El videostreaming es la tecnología que permite la retransmisión de archivos multimedia (audio o audio y video) a través de Internet. La diferencia principal con los métodos tradicionales es que para poder ver un video (o escuchar un audio) no hace falta descargarse el fichero entero en la computadora y, por lo tanto, esperar todo ese tiempo. Mediante el videostreaming, el servidor, previa demanda, comienza a enviarnos fragmentos del archivo en el mismo momento que lo solicitemos y a una velocidad acorde con el ancho de banda de nuestra conexión a Internet [Alicante].

4. Agregar modificaciones al programa en el que se acoplen nuevas herramientas que ofrece la librería de OpenCV y Python para un mejor procesamiento y detección de patrones dentro de la imagen para un mayor rango de visión y localización.

5. Ampliar el margen de proyectos en el que se haga uso de la herramienta de procesamiento digital que nos ofrece la librería de Opencv y Python

6. Basándonos en el prototipo realizado, implementar un software con comunicación inalámbrica que permita un control y manipulación de imágenes a distancia, logrando así también eliminar las limitantes de velocidad y de giro del dispositivo con respecto al peso y dimensiones del mismo.

Apéndice A

Código G

Para poder llevar a cabo el corte de las placas y las perforaciones de las mismas, se utilizó el lenguaje G [Teruel04]. El código en lenguaje G para la máquina CNC que realizó los cortes para las bases del robot se muestra a continuación:

```
G90 (programación en coordenadas absolutas)
G21 (trabajar con milímetros)
G00 Z20 (levantamos herramienta)

(HOYOS: Se procede primero a realizar los cortes internos y al final el contorno externo)
G00 X20 Y10
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G00 Z20 (levantar herramienta)

G00 X80 Y10
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G00 Z20 (levantar herramienta)

G00 X40 Y25
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G01 X40 Y60
G00 Z20 (levantar herramienta)

G00 X60 Y60
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G01 X60 Y25
G00 Z20 (levantar herramienta)

G00 X30 Y85
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G01 X70 Y85
G00 Z20 (levantar herramienta)

G00 X30 Y97
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G01 X70 Y97
G00 Z20 (levantar herramienta)

G00 X30 Y109
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
```

```
G01 X70 Y109
G00 Z20 (levantar herramienta)

G00 X10 Y120
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G00 Z20 (levantar herramienta)

G00 X90 Y120
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G00 Z20 (levantar herramienta)


(RECTANGULOS)

G00 X100 Y130
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G01 X100 Y85
G01 X85 Y85
G01 X85 Y15
G01 X100 Y15
G01 X100 Y10

(bisel)
G00 X100 Y10
G02 X90 Y0 I-10 J0 F250.0

G01 X10 Y0

(bisel)
G00 X10 Y0
G02 X0 Y10 I0 J10 F205.0

G01 Y15
G01 X15
G01 Y85
G01 X0
G01 X0 Y130

(ARCO)
G00 X0 Y130
G02 X100 Y130 I50 J0 F250.0

G00 Z20 (levantar herramienta)

M30
```

Apéndice B

Código Controlador PID

Para poder controlar el movimiento de los giros del robot móvil se utilizó el siguiente código, el cual corresponde a un controlador PID que fue el que se utilizó en el desarrollo de este proyecto.

```
#include "math.h"
#define dirM1_1 13
#define dirM1_2 11
#define dirM2_1 10
#define dirM2_2 9
float error=0, error_anterior, proporcional, derivativo, integral;
float Kp=(float)1/10000, Ki=0.00, Kd=(float)1.20;
int vel_motor1 =95;
int vel_motor2 =95;
int vel_base = 95;
int referencia = 0;
void setup() {
pinMode(dirM1_1, OUTPUT);
pinMode(dirM1_2, OUTPUT);
pinMode(dirM2_1, OUTPUT);
pinMode(dirM2_2, OUTPUT);
  analogWrite(dirM1_1,70);
  analogWrite(dirM2_1,70);
  Serial.begin(9600);
}
int leerSensores(){
  int distanciaXcYc;
  char signo;
  while(!Serial);
  distanciaXcYc = Serial.parseInt();
  return distanciaXcYc;
}
void adelante(int vel_motor1, int vel_motor2){
  analogWrite(dirM1_1,vel_motor1);
  digitalWrite(dirM1_2,LOW);
  analogWrite(dirM2_1,vel_motor2);
  digitalWrite(dirM2_2,LOW);
}
```

```
void loop() {
  float actual = leerSensores();
  if(actual <= -160) actual = 100;
  error_anterior=error;
  error = actual - referencia;
  proporcional = (float)Kp * error;
  integral += Ki*error;
  derivativo = Kd*error - error_anterior;
  float cpid = proporcional + integral + derivativo;
  if(cpid < 0){
     vel_motor1 = vel_base - cpid;
     vel_motor2 = vel_base + cpid;
  } else {
     vel_motor1 = vel_base - cpid;
     vel_motor2 = vel_base + cpid;
  }
if(vel_motor1 > vel_base ) vel_motor1=vel_base;
if(vel_motor1 < 0 ) vel_motor1=30;
if(vel_motor2 > vel_base ) vel_motor2=vel_base;
if(vel_motor2 < 0 ) vel_motor2=30;
  adelante(vel_motor1,vel_motor2);
  delay(10);
}
```

Apéndice C

Sistema de visión de prueba

El objetivo de este código es poder realizar pruebas en una computadora personal del sistema de visión computacional. Se llevó a cabo esta implementación tomando la lectura de video a partir de un archivo, permitiendo identificar el ajuste requerido de parámetros bajo diferentes condiciones ambientales, después de realizar algunas pruebas y ajustes se colocó el sistema final sobre la Raspberry pi con los cambios indicados en el apéndice D. El código implementado se muestra a continuación:

```
import numpy as np
import cv2
import time

class Linea():

def ubicacion(self,x,cols):
    centrox = x-(cols/2)
     return centrox
def __init__(self):
pass
def EncuentraContornoMayor(self,VarContornos):
     maxp=0
     cntmax=np.ones((1,1),np.uint8)
     for cnt in  VarContornos:
         perimetro=cv2.arcLength(cnt,False)
         if perimetro>maxp:
             maxp=perimetro
             cntmax=cnt
     return cntmax
def principal(self):
xanterior=0
rango =10
Hmin=0
Hmax=115
Smin=0
Smax=89
Vmin=0
Vmax=80
```

```
cap = cv2.VideoCapture('output.avi')
cols = cap.get(3)
rows = cap.get(4)
centrox=0
print (cols,rows)

# Iniciamos el bucle de captura, en el que leemos cada frame de la captura

while(True):
    ret, frame = cap.read()
    time.sleep(0.09)

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

                    roiS = hsv[:rows/2,:cols]
    roiI = hsv[rows/2:,:cols]

    roiIf = frame[rows/2:,:cols]
    roiSf = frame[rows/2:,:cols]

    inferior = cv2.inRange(roiI, np.array((Hmin,Smin,Vmin)), np.array((Hmax,Vmax,Smax)))
    superior = cv2.inRange(roiS, np.array((Hmin,Smin,Vmin)), np.array((Hmax,Vmax,Smax)))
    inferior = cv2.erode (inferior,cv2.getStructuringElement(cv2.MORPH_RECT,(2,2)),iterations = 2)
    inferior = cv2.dilate (inferior,cv2.getStructuringElement(cv2.MORPH_RECT,(4,4)),iterations = 4)

    superior = cv2.erode (superior,cv2.getStructuringElement(cv2.MORPH_RECT,(2,2)),iterations = 2)
    superior = cv2.dilate (superior,cv2.getStructuringElement(cv2.MORPH_RECT,(4,4)),iterations = 4)

    contornosI,jerarquia = cv2.findContours(inferior,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    cntmaxI = self.EncuentraContornoMayor(contornosI)

    momentos = cv2.moments(cntmaxI,0)

    cx=0
    cy=0
    # Localizamos la posicion del objeto
    M = cv2.moments(inferior)
    xanterior=centrox
    if M['m00']>50000:
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
    centrox = cx-(cols/2)
    if(centrox==-(cols/2)):
print 'centrox'
print (centrox)
centrox=xanterior
print 'anterior'
print (centrox)
    if self.ubicacion(cx,cols) <0:
     print 'd'
    elif self.ubicacion(cx,cols)>0:
     print 'i'
    else:
     print 'c'

    self.ubicacion(cx,cols)
    cv2.rectangle(roiIf,(cx-rango,cy+rango),(cx+rango,cy-rango),(0,255,0),2)
    cv2.rectangle(inferior,(cx-rango,cy+rango),(cx+rango,cy-rango),(255,255,255),2)
    # Creamos las ventanas de salida y configuracion
    cv2.imshow('Salida', frame)
    cv2.imshow('inRange', inferior)
    cv2.imshow('Superior', Superior)
    if cv2.waitKey(1) & 0xFF == ord('q'): # Indicamos que al pulsar "q" el programa se cierre
```

```
        break

cap.release()
cv2.destroyAllWindows()
l = Linea()
l.principal()
```

Apéndice D

Código de Sistema de Visión

El código del sistema implementado sobre la SBC se basa en el código de prueba del apéndice C. Los cambios importantes se basan en:

- La parte en donde se define el dispositivo de captura del video.
- Las instrucciones para leer los frames del dispositivo de captura

Las diferencias son debidas al driver implementado especialmente para la cámara de la Raspberry Pi como ya fue mencionado anteriormente.

```
import io
import numpy as np
import cv2
import time
import serial
import picamera
ser = serial.Serial('/dev/ttyACM0', 9600) // Puerto de comunicacion serial con la tarjeta de desarrollo Arduino

class Linea():
def ubicacion(self,x,cols):
    centrox = x-(cols/2)
     return centrox
def __init__(self):
pass
def EncuentraContornoMayor(self,VarContornos):
     maxp=0
     cntmax=np.ones((1,1),np.uint8)
     for cnt in  VarContornos:
         perimetro=cv2.arcLength(cnt,False)
         if perimetro>maxp:
             maxp=perimetro
             cntmax=cnt
     return cntmax
def principal(self):
xanterior=0
centrox=0
```

```
rango =10
Hmin=0
Hmax=115
Smin=0
Smax=89
Vmin=0
Vmax=80
cols = 160
rows = 140
print (cols,rows)
fourcc = cv2.cv.CV_FOURCC(*'XVID')
salidaf = cv2.VideoWriter('salida4f160x140.avi',fourcc,30.0,(cols,rows))
# Iniciamos el bucle de captura, en el que leemos cada frame del dispositivo
with picamera.PiCamera() as camera:
       stream = io.BytesIO()
       camera.resolution = (cols, rows)
       for foo in camera.capture_continuous(stream,format='jpeg',use_video_port=True):
    data = np.fromstring(foo.getvalue(), dtype=np.uint8)
    frame = cv2.imdecode(data, 1)
         hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) #Convertimos imagen a HSV
     roiI = hsv[rows/2:,:cols]
    roiIf = frame[rows/2:,:cols]

    inferior = cv2.inRange(roiI, np.array((Hmin,Smin,Vmin)), np.array((Hmax,Vmax,Smax)))

    # Limpiamos la imagen de imperfecciones con los filtros erode y dilate
    inferior = cv2.erode (inferior,cv2.getStructuringElement(cv2.MORPH_RECT,(2,2)),iterations = 2)
    inferior = cv2.dilate (inferior,cv2.getStructuringElement(cv2.MORPH_RECT,(4,4)),iterations = 4)
    contornosI,jerarquia = cv2.findContours(inferior,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    cntmaxI = self.EncuentraContornoMayor(contornosI)

    momentos = cv2.moments(cntmaxI,0)

    cx=0
    cy=0
    # Localizamos la posicion del objeto
    M = cv2.moments(inferior)
    print M['m00']
    xanterior=centrox
    if M['m00']>10000:
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
    centrox = cx-(cols/2)
    print "xc=\%d yc=\%d "\%(centrox,cy)
    if(centrox==-(cols/2)):
centrox=xanterior
if(centrox<0):
centrox=-cols/2
if(centrox>0):
centrox=cols/2
    signo='\n'
    ser.write(signo)
    ser.write(str(int(centrox)).encode())
    print centrox
    cv2.rectangle(roiIf,(cx-rango,cy+rango),(cx+rango,cy-rango),(0,255,0),2)
    cv2.rectangle(inferior,(cx-rango,cy+rango),(cx+rango,cy-rango),(255,255,255),2)
    # Creamos las ventanas de salida y configuracion
    cv2.imshow('Salida', frame)
    cv2.imshow('inRange', inferior)
    salidaf.write(frame)
    if cv2.waitKey(1) & 0xFF == ord('q'): \# Indicamos que al pulsar "q" el programa se cierre
        break
    stream.truncate()
```

```
    stream.seek(0)

cap.release()
cv2.destroyAllWindows()
l = Linea()
l.principal()
```

Referencias

[Alicante] Alicante, U. D. Videostreaming. http://aplicacionesua.cpd.ua.es/videostreaming/introduccion.asp. Consulta: 28 de Abril de 2015.

[APL] Aplicación de la robótica. http://www.industriaynegocios.cl/Academicos/AlexanderBorger/Docts\%20Docencia/Seminario\%20de\%20Aut/trabajos/2004/Rob\%C3\%B3tica/seminario\%202004\%20robotica/Seminario_Robotica/Documentos/APLICACI\%C3\%93N\%20DE\%20LA\%20ROB\%C3\%93TICA.htm.

[Arduino08] Arduino. Arduino nano (v2.3) user manual. https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf, 2008. Consulta: Abril 2015.

[Baturone01] Baturone, A. *Robotica : manipuladores y robots moviles*. Marcombo Alfaomega, Barcelona, 2001. ISBN 8426713130.

[Bradski08] Bradski, G. y Kaehler, A. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media,Inc., 2008.

[Corporation15] Corporation, I., Garage, W., y Itseez. Opencv. https://en.wikipedia.org/wiki/OpenCV, 2015.

[GCO] Emc2 g-code. http://linuxcnc.org/docs/2.1/html/gcode.html. Consulta: 1 de junio de 2015.

[globalsegurity.org] globalsegurity.org. http://www.globalsecurity.org/military/systems/ground/talo Consulta: 18 de agosto de 2015.

[Himax12] Himax. Hx8357-d00/d01, 320rgb x 480 dot, 16m color, tft mobile single chip driver. `http://www.adafruit.com/datasheets/HX8357-D_DS_April2012.pdf`, 2012.

[Instruments98] Instruments, T. Sn754410 quadruple half-h driver. `http://pdf1.alldatasheet.es/datasheet-pdf/view/28616/TI/SN754410NE.html`, 1998.

[Instruments03] Instruments, T. a7800 series positive-voltage regulators. `https://www.sparkfun.com/datasheets/Components/LM7805.pdf`, 2003.

[Jones14] Jones, D. Raspberry pi camera (python - picamera). `http://picamera.readthedocs.org/en/release-1.10/`, 2013, 2014.

[Laganiére11] Laganiére, R. *OpenCV 2 computer vision application programming cookbook over 50 recipes to master this library of programming functions for real-time computer vision*. Packt Pub, Birmingham, UK, 2011. ISBN 978-1849513241.

[limoncellodigital] limoncellodigital. http://www.limoncellodigital.com/2011/05/hasta-siempre-spirit.html. Consulta: 18 de agosto de 2015.

[Marr10] Marr, D. *Vision : a computational investigation into the human representation and processing of visual information*. MIT Press, Cambridge, Mass, 2010. ISBN 9780262514620.

[Martinsanz08] Martinsanz, G. P. *Visión por computador: Imágenes Digitales y Aplicaciones*. 2ª ed$^{\underline{\text{ón}}}$., 2008. ISBN 9788478978311.

[Mateos] Mateos, J. http://proton.ucting.udg.mx/dpto/tesis/quetzal/CAPITUL1.html. Consulta: 13 de mayo de 2015.

[Medina] Medina, E. Puente h. http://www.xatakaciencia.com/robotica/robots-moviles-i. Consulta: 28 De Mayo de 2015.

[Moisés14] Moisés, G. V., Silva C., J. C., y Torres R., A. Robot móvil autónomo seguidor de objetos con visión computacional. págs. 7–8, 2014.

[opencv] opencv. http://opencv.org/. Consulta: 23 de junio de 2015.

[Ortiz06] Ortiz, J. A. R. Desarrollo de una aplicación de inspección visual, utilizando visión por computadora. pág. 2, 2006.

[roboticspot] roboticspot. http://www.roboticspot.com/robots.php?id=11. Consulta: 18 de agosto de 2015.

[Sebastian12] Sebastian, C. Regulador 78xx ¿2a. `http://electgpl.blogspot.mx/2012/05/regulador-78xx-2a.html`, 2012.

[Suzuki85] Suzuki, S. et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[Teruel04] Teruel, F. C. *Control numérico y programación*. MARCOMBO, 2004. ISBN 8426713599.
URL `http://www.amazon.com/Control-num\%C3\%A9rico-programaci\%C3\%B3n-Francisco-Teruel/dp/8426713599\%3FSubscriptionId\%3D0JYN1NVW651KCA56C102\%26tag\%3Dtechkie-20\%26linkCode\%3Dxm2\%26camp\%3D2025\%26creative\%3D165953\%26creativeASIN\%3D8426713599`

[Upton12] Upton, E. *Raspberry Pi user guide*. Wiley, Chichester, West Sussex, UK, 2012. ISBN 978-1-118-46446-5.

[xatacaciencia] xatacaciencia. Robots móviles. http://blutintegrado.blogspot.mx/2012/05/puen
h.html. Consulta: 28 De Mayo de 2015.

[Åström09] Åström, K. J. y Hägglundm, T. *Control PID avanzado*. Pearson Educacion, Madrid España, 2009. ISBN 978-84-8322-511-0.
URL `http://cursos.itcg.edu.mx/libros/PID\%20avanzado.pdf`