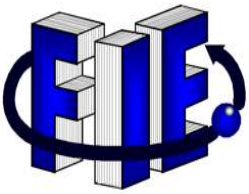

Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica

PROTOTIPO DE UN ROBOT MÓVIL AUTÓNOMO Y LA IDENTIFICACIÓN DE OBJETOS MEDIANTE VISIÓN COMPUTACIONAL

TESIS

Que para obtener el grado de
INGENIERO EN COMPUTACIÓN

Presenta

Alberto Torres Ramírez

-

Asesor de Tesis

M.I. Moisés García Villanueva

Morelia, Michoacán, Diciembre 2015

Agradecimientos

Quiero agradecer a la Facultad de Ingeniería Eléctrica de la UMSNH por el espacio brindado en sus aulas, así como a mi asesor: M.I. Moisés García Villanueva por los conocimientos brindados para la realización de este trabajo y la amistad cosechada a través de este largo tiempo, así como a mis padres Juan Torres Alvarado e Irma Ramírez Juárez por su apoyo, no sólo durante mi periodo de preparación universitario, sino durante toda la vida, así como a mi novia Samantha Alexandra León Padilla por su apoyo y comprensión en momentos difíciles y por su cariño y ternura en todo momento, a un gran amigo y colega Misael Armenta Nieto ya que, con él empezamos este desafío, que sin saberlo, sería sin duda alguna, una de las cosas más difíciles a la cual nos hemos enfrentado hasta ahora, me alegra saber que juntos entramos y juntos salimos de esta facultad, así como también agradezco a mis hermanos, amigos y personas que estuvieron a mi lado durante esta dura etapa de mi vida, pero que a la vez, fue una de las más gratificantes metas que he cumplido.

Dedicatoria

Dedico este trabajo a todas las personas que, de forma directa o indirecta formaron parte de este camino, a veces pedregoso, a veces, lleno de verde pasto, este camino llamado Universidad, que sin duda alguna parecía imposible llegar al final, varias veces me perdí en este camino, pero gracias al apoyo de algunas personas pude salir de ese momento difícil y seguir avanzando, dedico pues este trabajo a aquellas personas que tendieron su mano para ayudarme cuando caí, ya que gracias a ellos me levanté, así mismo dedico también este trabajo a las personas que cuando caía pasaban junto a mí, ignorándome, ya que si no fuera por ellos no hubiera tenido el valor para levantarme de nuevo y seguir adelante con más fuerza que antes, ya con eso comprendí, que lo que el árbol tiene de florido, vive de lo que tiene sepultado.

Resumen

El desarrollo tecnológico nos ha permitido llevar a cabo implementaciones de equipos o prototipos que sirven de apoyo en la didáctica, divulgación y autoaprendizaje. En este trabajo se presenta la implementación de un sistema de visión computacional que permite la identificación de objetos por medio del color, el cual es conjuntado con un prototipo de un robot móvil autónomo, resultando un robot de servicio.

La tarea que se le exige al robot es visualizar un objeto de un color definido, realizar el desplazamiento hacia él y tomarlo con sus pinzas. Se emplea un sistema de control PID (Proporcional, Integral, Derivativo) para controlar la velocidad de los motores con que cuenta el robot, de tal manera que pueda realizar el desplazamiento correcto hacia el objeto. El sistema de visión está compuesto por una cámara, que visualiza el ambiente, dicha información es emitida a una unidad de procesamiento (microcomputadora PCB Rapsberry Pi), en la que se determina mediante técnicas de visión computacional y las librerías de OpenCV, si el objeto es del color y forma deseados. Existe un mecanismo de comunicación entre la microcomputadora y la tarjeta de desarrollo Arduino, a través del puerto USB o emulación del puerto serie, que permiten enviar un valor de referencia al sistema de control de los motores que desplazan el robot. El robot cuenta además con un sensor infrarrojo de distancia que mide la cercanía a un objeto, momento en el cual se acciona el mecanismo que manipula la pinza que hace posible tomar o capturar el objeto. Las fases de diseño y armado del prototipo se pueden dividir en 3 categorías principales: circuitería electrónica, estructura física y programación. Se logró realizar la tarea planteada exitosamente, lo cual se demuestra en la parte experimental del presente trabajo.

Palabras clave:

Visión computacional, Robótica móvil, Control PID, Identificación de objetos, Seguimiento de objetos.

Abstract

Technological development has allowed us to carry out implementations of equipment or prototypes that support the teaching, dissemination and self-learning. This paper implementing a computer vision system allowing the identification of objects by means of color, which is outfitted with a prototype of an autonomous mobile robot, resulting in a service robot.

The task for the robot is to search and allocate an object of a defined color, make the shift to him and take it with tweezers. Control system PID (Proportional, Integral, Derivative) is used to control the speed of motors available on the robot, so that can perform the correct offset at the object. The vision system consists of a camera, which senses the environment, this information is issued to a processing unit (microcomputer PCB Rapsberry Pi), which use computer vision techniques and libraries OpenCV, if the object has the desired color and shape. There is a mechanism of communication between the microcomputer and the Arduino development board, USB port or serial port emulation, which allow it to send a reference value to the control system of the motors that move the robot. The robot also has an infrared distance sensor that measures the distance to a object, at which the mechanism that manipulates the clamp makes possible to take or capture the object. The phases of design and assembly of the prototype can be divided into 3 main categories: electronic circuitry, physical structure and programming. We accomplished the task successfully, which is demonstrated in the experimental part of this work.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

CV	**C**omputer **V**ision (Visión computacional)
VGA	**V**ideo **G**raphics **A**rray (Arreglo o adaptador gráfico de video)
PX/px	**P**ixeles
FPS	**F**rames **P**er **S**econd (Cuadros o imágenes por segundo)
RGB	**R**ed **G**reen **B**lue (Rojo, Verde y Azul
PID	**P**roporcional **I**ntegral **D**erivativo
PWM	**P**ulse **W**idth **M**odulation (Modulacion por Ancho de Pulsos)
HSV	**P**roporcional **I**ntegral **D**erivativo
HSV	**H**ue **S**aturation **V**alue (Matiz, Saturación y Valor)
HSI	**H**ue **S**aturation **I**ntensity (Matiz, Saturación e Intensidad)
IARP	**I**nternational **A**dvanced **R**obotics **P**rogramme (Programa Internacional en Robótica Avanzada)
SRI	**S**tanford **R**esearch **I**nstitute (Instituto de Investigaciones de Stanford)

Capítulo 1

Introducción

1.1. Antecedentes

Aunque el entorno industrial sigue siendo el campo de aplicación por excelencia de la Robótica, desde hace más de dos décadas han ido surgiendo aplicaciones alternativas cuyo interés va en aumento en los últimos años con la aparición de los primeros productos comerciales. Esto ha hecho que paulatinamente se vayan popularizando términos como "robótica avanzada" o "robots de servicio". Una de las primeras actuaciones significativas en esta línea fue el lanzamiento del proyecto internacional de colaboración denominado en la actualidad "Programa Internacional en Robótica Avanzada" (IARP, por sus siglas en Inglés de International Advanced Robotics Programme) iniciado en 1982, a raíz de la Cumbre Económica de Versalles, con el objetivo de potenciar la colaboración internacional para el desarrollo de sistemas robotizados que permitan reducir la participación humana en actividades complejas en entornos o condiciones inhóspitos, agresivos o peligrosos [Marinero02].

Hoy en día las empresas en general, han ido aumentando en cantidad su nivel de producción. Uno de los factores se debe a la creciente población, de tal forma que va en aumento la demanda de los productos que se ofrecen; este incremento en la demanda implica que el trabajo de gestión de control de calidad sea más arduo y por ende las empresas requieren de sistemas automáticos que les ayuden a reducir costos de operación, impactando ello de forma monetaria, en la calidad de los productos y del tiempo en el proceso productivo

[Siegwart11].

Si bien en la actualidad los robots industriales realizan múltiples tareas con gran precisión y rapidez, el principal hándicap que presentan está en la falta de movilidad de los mismos, pudiendo sólo actuar dentro de la zona su trabajo, entendiendo ésta como el volumen del espacio definido por los puntos accesibles por el elemento terminal del robot [Siegwart11].

Por otra parte, en el trabajo desarrollado por Scott E. Umbaugh [Umbaugh97] en el año de 1997, se señala que el área de procesamiento de señales ha sido objeto de estudio de ingenieros computacionales, mientras que la manipulación de datos de las imágenes ha sido manejada por científicos computacionales; la convergencia de estas dos especialidades nos da como resultado el área de la visión computacional y procesamiento de imágenes.

Hoy en día, una técnica muy utilizada en las empresas, es la visión computacional, ya que las bibliotecas del software OpenCV contiene diversas funciones para varios propósitos, ya sea en la identificación de objetos, colores, contornos y muchas otras operaciones. En la robótica móvil, esta área de visión computacional, es la que permite que los robots visualicen el ambiente en el que se encuentran mediante cámaras, con las cuales perciben las formas y colores del espacio en el que se desenvuelven.

Los objetivos de la visión computacional, también conocida como visión artificial, incluyen entre otros, la detección, reconocimiento, identificación y localización de objetos [Wikipedia14].

1.2. Justificación

En base a la creciente demanda industrial y a la automatización de procesos que requiere el mundo actual, se necesita cada vez más de sistemas que realicen tareas de forma autónoma y correcta, para la detección de objetos, para su manipulación industrial, en la clasificación por color, en la categorización por color, forma o tamaño de productos en procesos industriales o transportar ciertos objetos de un lugar a otro.

Al usar un sistema automatizado basado en visión computacional, se espera tener un mejor desempeño en las labores de detección de objetos, ya que un sistema automatizado

es más confiable, seguro y económico que un sistema similar desempeñado por una persona.

La robótica de servicios incluye todos aquellos aspectos en los que un sistema robotizado interacciona con seres humanos en el mismo espacio de trabajo. Esto implica que factores como la seguridad, interacción con el ser humano y un mayor grado de autonomía en entornos parcialmente estructurados tienen mucha más importancia que en los robots industriales. Los robots de servicios son dispositivos mecatrónicos, sensorizados y reprogramables y su participación es beneficiosa en las actividades humanas cotidianas [Marinero02].

1.3. Objetivo de la tesis

1.3.1. Objetivo general

Desarrollar un sistema de identificación de objetos mediante visión computacional, el cual deberá distinguir objetos de diferentes colores; implementar un prototipo de un robot móvil que utilice el sistema de visión computacional para identificar los objetos, transportarlos y colocarlos en un depósito, tarea para la cual es diseñado el robot móvil autónomo.

1.3.2. Objetivos particulares

- Desarrollo e implementación del hardware del robot móvil.
- Adquisición del vídeo en tiempo real.
- Segmentación de una imagen mediante técnicas de color que identifican ROI's (Regiones de Interés).
- Seguimiento del objeto identificado mediante un sistema de control PID (por sus siglas en inglés de Proportional, Integral and Derivative).
- Tomar el objeto y ubicar el depósito.
- Transportar el objeto hacia el depósito y realizar una nueva búsqueda de objeto.
- Pruebas con diferentes objetos de color.

1.4. Descripción de los capítulos

- En el capítulo uno se da una breve introducción a este trabajo. Se mencionan antecedentes, se plantea el problema y se establecen los objetivos principales y particulares de la tesis. Se señala además una descripción de cada capítulo.

- El capítulo dos describe el marco teórico. Da una breve descripción de los términos a utilizar en este documento y se abordan los temas de la robótica móvil, visión computacional y robots de servicio.

- El capítulo tres describe el hardware y desarrollo del robot móvil autónomo de servicio para la tarea especificada.

- El capítulo cuatro describe el software y la implementación del sistema de visión computacional para la detección de objetos, aplicado en el robot de servicio.

- El capítulo cinco se refiere a los resultados y experimentación que se obtuvieron probando con los diferentes objetos, escenarios, así como diferentes espacios de color.

- El capítulo seis muestra las conclusiones, recomendaciones y propuestas para trabajos futuros.

Capítulo 2

Visión computacional y los robots de servicio

2.1. Introducción

Con el fin de tener un panorama de la importancia que ha tenido la robótica móvil a lo largo de estos años, se hace mención de los antecedentes históricos de ésta en el mundo. Se inicia en el año 1948 con William G. Walter, quien diseñó y construyó los primeros robots electrónicos móviles autónomos de ruedas. Sin embargo, no fue hasta 1970 con el robot Shakey, diseñado dentro de los laboratorios de inteligencia artificial del SRI (por sus siglas en inglés de Stanford Research Institute), que se incorporó la habilidad de reconocer su entorno por medio de sensores y una cámara de visión que llevaba integrada, este robot es considerado el pionero en lo que actualmente se conoce como inteligencia artificial [Cristiany12]. La figura 2.1 nos muestra un ejemplo de robot de servicio y el robot Shakey, pionero en los robots de servicio.

Según Aristóteles, Visión es saber qué hay y dónde mediante la vista, lo cual es esencialmente válido. Nuestra vista y cerebro identifican, a partir de la información que llega a nuestros ojos, los objetos que nos interesan y su posición en el ambiente, lo cual es muy importante para muchas de nuestras actividades [Sucar11].

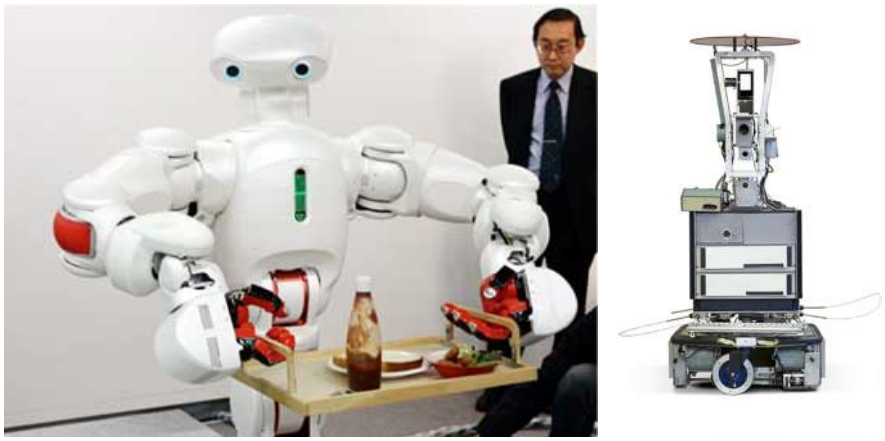

Figura 2.1: Se muestra en el lado izquierdo un robot de servicio actual [Manager14]; en el lado derecho el robot Shakey que es el pionero en este tipo de robots de servicio [Museum15].

2.1.1. Visión computacional

La Visión Computacional trata de alguna forma de emular el sentido de la vista con que contamos los humanos, en las computadoras; de tal forma que mediante la interpretación de las imágenes adquiridas, por ejemplo con una cámara, se puedan reconocer los diversos objetos en el ambiente y su posición en el espacio. La facilidad con la que "vemos" llevó a pensar a los primeros investigadores en inteligencia artificial, por 1960, que hacer que una computadora interpretara imágenes era relativamente fácil, pero no resultó así y muchos años de investigación han demostrado que es un problema muy complejo. La figura **??** muestra el uso de un sistema de visión computacional para la detección de personas [Sucar11].

Sin embargo, en los últimos años hay avances considerables básicamente por 3 factores:

- El desarrollo tecnológico en las capacidades de procesamiento y de memoria en las computadoras, que facilita el almacenamiento y procesamiento de las imágenes.

- Los avances teóricos en los principios y algoritmos para el procesamiento y análisis de imágenes.

- La creciente necesidad del procesamiento automático de imágenes, que se capturan y

almacenan en grandes cantidades en diversos dominios, como en medicina, seguridad, tránsito de vehículos, entre otros [Sucar11].

2.1.2. Visión computacional en la robótica móvil

El objetivo del sistema de percepción para el robot, sobre la base de sensores de visión, es transformar la imagen proporcionada por la cámara, en una descripción de los elementos presentes en el entorno del robot. Dicha descripción debe tener información necesaria para que el robot efectúe los movimientos que permitirán la ejecución de la tarea programada [Malpartida03]. Para alcanzar estos objetivos, se deberá elaborar las siguientes funciones:

- Iluminación de la escena a capturar.
- Captación de la imagen del entorno significativo del robot, conversión analógico/digital y adquisición por una computadora.
- Mejoramiento de la imagen y realzado de las características geométricas relevantes desde el punto de vista de la aplicación.
- Segmentación de la imagen.
- Descripción de la imagen y/o extracción de características.
- Reconocimiento de los objetos.
- Elaboración de las consignas para el sistema de control del robot.

La visión computacional desempeña un papel fundamental en el campo de la robótica móvil, dada la gran cantidad de información sobre el entorno, que puede ser extraída de una imagen. La robótica es una disciplina en la que se combinan diversas áreas de conocimiento como la mecánica, electrónica, informática, inteligencia artificial, ingeniería de control, cuyo principal objetivo es la automatización de procesos para la reducción de costos y/o mejora de las condiciones laborales [Pardo12].

Dentro de la robótica se encuentra la robótica móvil, disciplina que se centra en el estudio de la movilidad de los robots. Dentro de la robotica la mayor parte de las aplicaciones

se ubican en la teleoperación de robots (movimientos controlados por un operador humano), o bien en el comportamiento autónomo de los mismos (sin ningún tipo de control por parte de los seres humanos)[Santonja00].

El IARP (por sus siglas en Inglés de International Advanced Robotics Programme) no proporciona una definición explícita de "robot avanzado". En su lugar plantea que éstos deben tener una o más de las siguientes características:

- Ser capaces de desplazarse de forma autónoma.
- Poder operar en condiciones o entornos inhóspitos, agresivos o peligrosos.
- Ser utilizados en aplicaciones no industriales: Agricultura, nuclear, espacio, medicina, Ingeniería civil y construcción, operaciones de rescate, fabricación inteligente, entre otros [Marinero02].

Otra muestra de la creciente importancia de este sector es la creación del Comité Técnico sobre Robots de Servicios IEEE Robotics and Automation Societ, durante la Conferencia Internacional sobre Robótica y Automatización de 1995 en Nagoya, con el objetivo de servir de apoyo a la creciente comunidad internacional de investigadores e ingenieros que desarrollan sistemas robotizados que prestan servicios de diferentes clases al ser humano.

La IFR (por sus siglas en Inglés de International Federation of Robotics) ha elaborado la siguiente definición preliminar de robot de servicios: "Un robot que funciona de forma total o parcialmente autónoma para proporcionar servicios de utilidad del ser humano o en instalaciones y equipos, excluyendo las operaciones de fabricación". Este mismo organismo propone la siguiente clasificación:

- Servicio al ser humano: entretenimiento, atención sanitaria, etc.
- Servicios aplicables a instalaciones o equipos: mantenimiento, reparación, limpieza, etc.
- Otras funciones realizadas de forma autónoma (vigilancia, transporte, toma de datos) y/o robots de servicio no clasificables en las dos categorías previas.

Otras definiciones propuestas son: La robótica de servicios incluye todos aquellos aspectos en los que un sistema robotizado interacciona con seres humanos en el mismo espacio de trabajo. Esto implica que factores como la seguridad, interacción con el ser humano y un mayor grado de autonomía en entornos parcialmente estructurados tienen mucha más importancia que en los robots industriales. Los robots de servicios son dispositivos mecatrónicos, sensorizados y reprogramables cuya participación es beneficiosa en las actividades humanas cotidianas. Se sitúan en algún punto entre los robots industriales y los robots de campo [Marinero02]. La figura 2.2 muestra la cronología y evolución del robot ASIMO.

Figura 2.2: Cronología y evolución del robot ASIMO, producto de la compañía Honda.

2.2. Sectores de aplicación

En la actualidad este tipo de robots móviles con visión computacional se pueden utilizar en varios sectores, ya sea en el sector industrial para clasificar o acomodar objetos, así como también en empresas. Uno de los casos más sobresalientes de robots con visión computacional es el robot ASIMO, diseñado por la compañía Honda. Es uno de los robots más avanzados que actualmente existen, se utiliza como robot de servicios, ya sea como guía dentro de la empresa o para hacer labores domésticas.

2.3. ASIMO: un robot con Sistema de Visión Computacional

En 1986, los ingenieros de Honda se propusieron crear un robot andante. Los primeros modelos (E1, E2, E3) se centraron en el desarrollo de las piernas que pueden simular el andar de un ser humano. La siguiente serie de modelos (E4, E5, E6) se centró en la estabilización a pie y subir escaleras. A continuación, se añadieron una cabeza, el cuerpo y los brazos al robot para mejorar el equilibrio y añadir funcionalidad. El primer robot humanoide de Honda, P1 fue bastante accidentado, tenia una altura de 6'2" y un peso de 386 libras. El siguiente modelo, P2 es una versión con un diseño más amigable, mejora la función de caminar, subir y bajar escaleras, incorpora los movimientos automáticos inalámbricos. El modelo P3 fue aún más compacto, con una altura de 5' 2" y un peso de 287 libras.

ASIMO es la culminación de dos décadas de investigación robótica humanoide por los ingenieros de Honda; puede correr, caminar en pendientes y superficies irregulares, girar suavemente, subir escaleras, y tratar de alcanzar y agarrar objetos, así como también puede comprender y responder a los comandos de voz simples. Tiene la capacidad de reconocer la cara de un selecto grupo de individuos. Usando cámaras, puede reconocer su entorno y registrar objetos inmóviles, evitar los obstáculos en movimiento mientras se mueve a través de su medio ambiente. Dado que el desarrollo continúa, Honda hoy muestra al robot a todo el mundo para alentar e inspirar a los jóvenes estudiantes a estudiar las ciencias. ASIMO también puede realizar ciertas tareas que son peligrosas para los seres humanos, tales como la lucha contra los incendios o la limpieza de los derrames tóxicos. La figura 2.3 nos muestra las diferentes vistas del robot de servicio ASIMO [Hirose07].

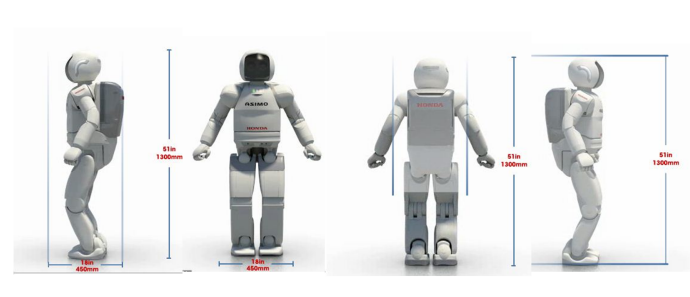

Figura 2.3: Robot de servicio ASIMO.

Capítulo 3

Hardware del robot

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

3.1. Raspberry Pi

Raspberry Pi es una computadora de placa reducida o placa única de bajo costo (SBC por sus 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. Es posible navegar en internet, enviar correos o escribir en un procesador de textos y además de realizar prácticas de programación [Upton12]. Actualmente se encuentra disponible una nueva versión de la Raspberry Pi, con mejores características que las descritas para la versión B. Durante el desarrollo del presente trabajo se tiene disponible la versión B de la Raspberry Pi.

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

Figura 3.1: Raspberry Pi.

3.2. 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 3.2, 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 de la Raspberry Pi.
- Módulo de cámara 5MP omnivisión 5647.
- Resolución de imagen: 2592 x 1944.
- Vídeo: 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 3.2: Camara Pi diseñada para la conexión en un puerto SCI de la Raspberry Pi.

3.3. 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 3.3. 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 3.3 [Wikipedia15].

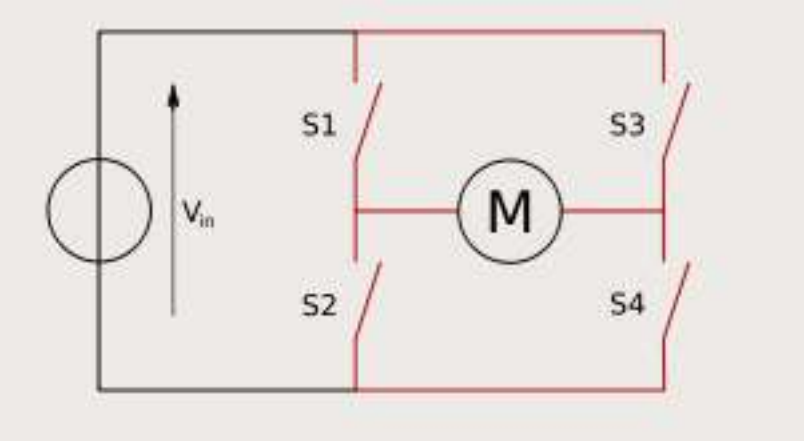

Figura 3.3: Puente H representado mediante interruptores.

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 3.4. 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 continua.
- Frecuencia de operación: 0 a 1 MHz.

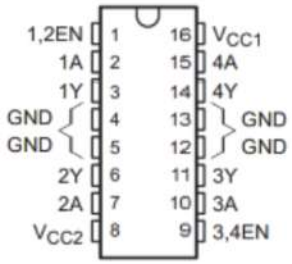

Figura 3.4: Disposición de los pines del puente H SN754410NE.

En la tabla 3.1 se especifican los pines del dispositivo SN754410NE mostrado en la figura 3.4.

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	Alimentación 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 3.1: Pines del SN754410NE.

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 pines de habilitación son conectados directamente a una señal digital en alto (Vcc 5 Volts) como lo muestra en la figura 3.5, 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.

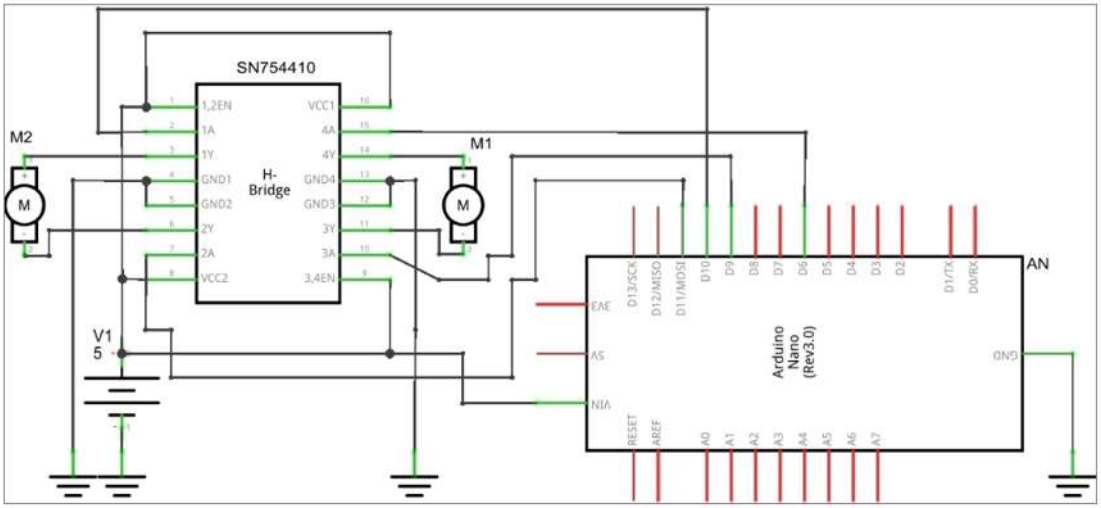

Figura 3.5: Diagrama de conexión del circuito integrado SN754410NE (puente H) y la tarjeta Arduino Nano.

3.4. 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 3.2.

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

Tabla 3.2: Características de los motorreductores.

En la figura 3.6 se muestra la imagen de los motores utilizados para este proyecto.

Figura 3.6: Motorreductores.

3.5. Servomotor

Un Servo es un dispositivo pequeño que tiene un eje de rendimiento controlado. Éste puede ser llevado a posiciones angulares específicas al enviar una señal codificada. Con tal de que una señal codificada exista en la línea de entrada, el servo mantendrá la posición angular del engranaje. Cuando la señal codificada cambia, la posición angular de los piñones cambia [Mancha15]. Los servomotores se utilizan para realizar la acción de abrir y cerrar las pinzas del robot para tomar el objeto, una vista de un servomotor se muestra en la figura 3.7.

Figura 3.7: Servomotor.

Características:

- Torque(4.8V): 2.7 kg-cm (37.5 oz/in).
- Torque(6.0V): 3.0 kg-cm (41.7 oz/in).

- Velocidad: 0.14 sec (4.8V). 0.12 sec (6.0V).
- Voltaje Operativo: 4.8 a 6.0 DC Volts.
- Peso: 16.0 g (0.56 oz).
- Tamaño:28.0 x 13.2 x 30.2 mm
- Frecuencia de trabajo: 1520 microsegundos / 50hz

3.6. 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 utilizarán 2 ruedas, las cuales serán giradas por un motor cada una, además de una rueda loca que permite direccionar libremente al robot, así que ello facilita sus giros. En la figura 3.8 se observan las llantas del robot que se usaron para el armado del mismo.

Figura 3.8: Ruedas del robot.

3.7. Rueda loca

La rueda loca es una variedad especial de rodamiento, ya que no tiene ningún eje fijo de giro y ésto provoca que pueda girar hacia cualquier dirección, también existe en varios materiales como es metal y plástico. En la figura 3.9 se muestra dicha rueda.

Figura 3.9: Rueda loca del robot.

3.8. Sensor infrarrojo

El sensor es un dispositivo electrónico/mecánico/químico que mapea un atributo ambiental resultando una medida cuantizada, normalmente un nivel de tensión eléctrica. Existen varios tipos de sensores de distancia, los más populares son: infrarrojos y ultrasónicos. Lo sensores infrarrojos son sensores analógicos, que envían un haz de luz si existe un objeto a una determinada distancia, el haz de luz rebota en el objeto y regresa de nuevo al sensor a través del receptor, de tal forma que con ello se percibe que existe un objeto. El funcionamiento del sensor ultrasónico es similar, solo que en vez de emitir luz emite sonido a una determinada frecuencia y la distancia a la que detecta el objeto es mayor. El sensor utilizado en este prototipo es el siguiente: SHARP GP2Y0A41SK0F de 4-30 cm. Es un sensor infrarrojo analógico muy utilizado en minirobótica para la detección de objetos u obstáculos en robots sumos o resuelve laberintos. Tanto el emisor como el receptor están encapsulados en un empaque, con la ventaja de que ya están alineados [Kú15]. La figura 3.10 muestra el sensor de distancia.

Figura 3.10: Sensor de distancia SHARP de 4-30 cm.

Las características de dicho sensor son las siguientes:

- Voltaje de operación: 4.5 V a 5.5 V.
- Consumo de corriente: 12 mA.
- Rango de distancia: 4 cm a 30 cm.
- Tipo de salida: Voltaje analógico.
- Periodo de actualización: 16 ms
- Tamaño: 29.5 x 13.0 x 13.5 mm.
- Peso: 3.5 g (0.12 oz)

La gráfica de operación del sensor se muestra en la figura 3.11 [Kú15].

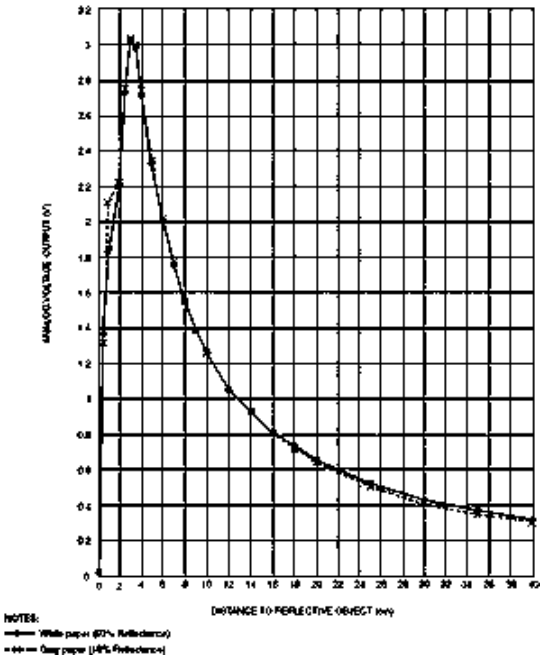

Figura 3.11: Gráfica del sensor de distancia SHARP 4-30 cm.

3.9. 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. Dicho microcontrolador 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 3.3:

En la figura 3.12 se muestra el Arduino Nano utilizado en el robot móvil.

Figura 3.12: Arduino Nano.

Para mayor detalle de las especificaciones técnicas y funcionalidad de la tarjeta

Elemento	**Descripción**
Microcontrolador	Atmel ATmega168 o ATmega328
Voltaje de Operación (nivel lógico)	5 Volts
Voltaje de entrada (recomendado)	7,12 Volts
Voltaje de entrada (límites)	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 3.3: Tabla de especificaciones de Arduino.

dirigirse al manual de usuario del Arduino Nano [Arduino08].

3.10. 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 3.4.

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

Tabla 3.4: 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 3.13 nos muestra el tipo de batería 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 3.13: Batería lipo a 11.4 volts del robot.

3.11. Fuente reguladora de voltaje

Para el suministro de energía (5 volts) de la SBC (Raspberry 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 3.14 [Sebastian12], en donde también se muestra la disposición de los dispositivos en una vista tridimensional.

3.12. 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.

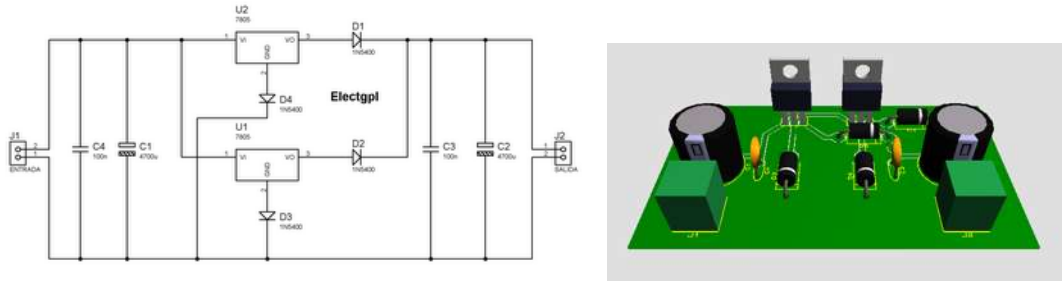

Figura 3.14: Fuente regulada de voltaje a 5 volts 2A.

Se integró al robot una pantalla táctil de la marca Adafruit de 3.5", ver la figura 3.15, 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 3.5.

Figura 3.15: 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 3.5: Características de la pantalla táctil 3.5".

3.13. 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 en 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 órdenes 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, LinuxCNC15].

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 3.16, 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 3.16 se muestra el resultado del corte de las placas finales y en la figura 3.17 se muestran las placas reales con el corte realizado.

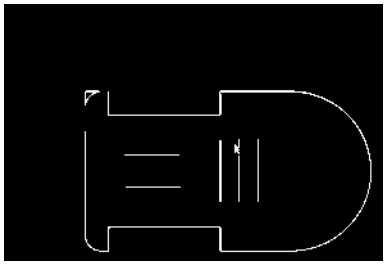

Figura 3.16: Dimensiones de la placa o base del robot móvil.

Figura 3.17: Placas del robot.

3.14. Pinzas del robot

El robot utiliza pinzas de acrílico para poder tomar los objetos, dichas pinzas también se diseñaron en la máquina de control numérico, están hechas del mismo material que las bases del robot, en la figura 3.18 se muestran las pinzas del robot.

Figura 3.18: Pinzas del robot.

En la figura 3.19 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.

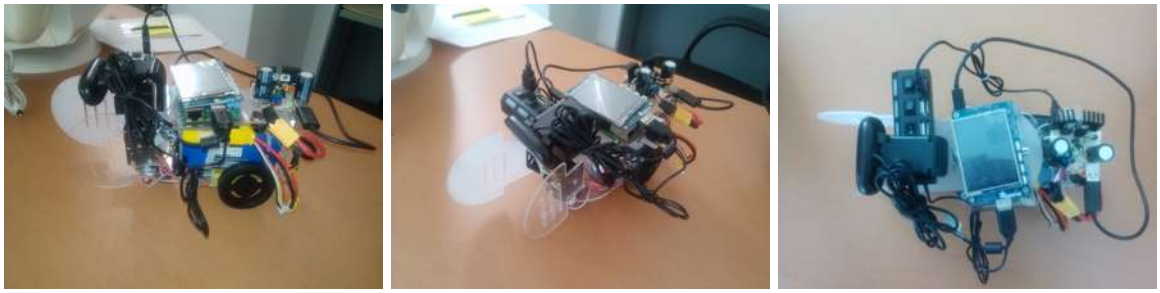

Figura 3.19: Figura del robot armado.

Finalmente en la figura 3.20 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.

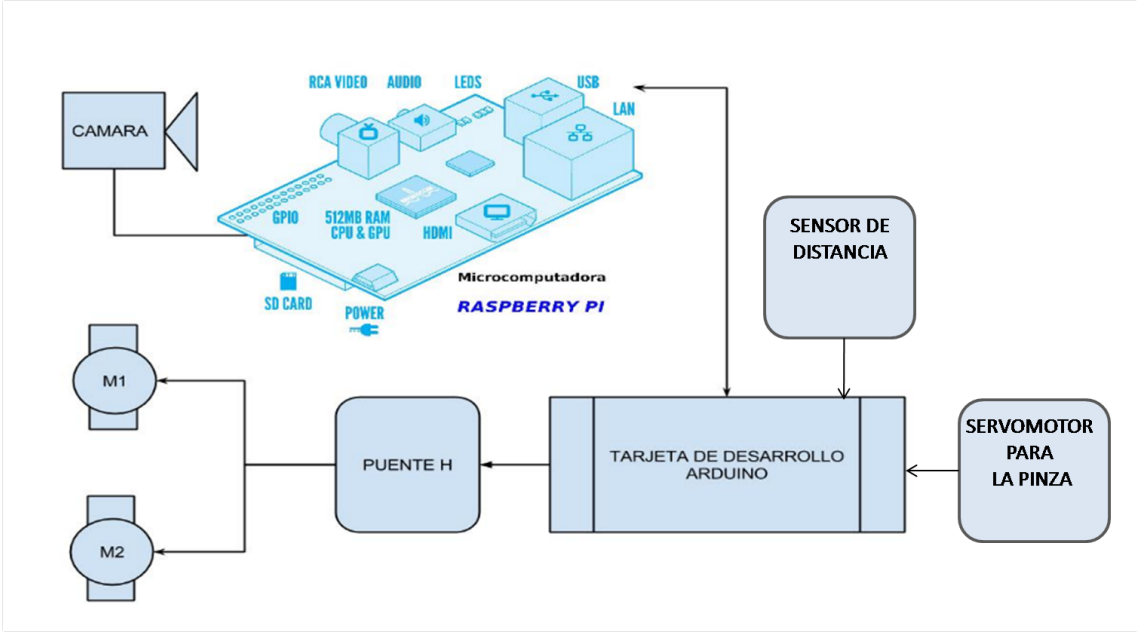

Figura 3.20: Diagrama general del sistema de visión con elemento de procesamiento, captura y bloques para el control del desplazamiento del robot.

Capítulo 4

Implementación del Software para la Identificación de Objetos

En este capítulo se describirá el software que se implementó para el sistema de visión computacional que permite la identificación y seguimiento de un objeto de color, y además la implementación del sistema de control de velocidad de los motores del robot de servicio. El sistema de visión del prototipo contempla las siguientes etapas:

1. Captura de información (vídeo).
2. Pre-procesamiento (mejora de la imagen mediante espacios de color).
3. Filtrado del objeto mediante técnica de color.
4. Segmentación (identificación de regiones de interés y objetos)
5. Seguimiento de objetos

En la figura 4.1 se muestra la forma en que se obtienen los píxeles que representan un objeto, los cuales permiten la identificación y seguimiento de éste en la escena. Primero se toma el vídeo, sobre los frames del vídeo se hacen las conversiones en espacio de colores con HSV (por sus siglas en Inglés Hue, Saturation and Value), HSI (por sus siglas en Inglés Hue, Saturation and Lightness) y BGR (por sus siglas en Inglés de Blue, Green and Red), posteriormente se filtran por el color del objeto, después se filtra para obtener los cúmulos

de píxeles, se obtiene el contorno del cúmulo de puntos y se obtiene el área, perímetro o centro de masa.

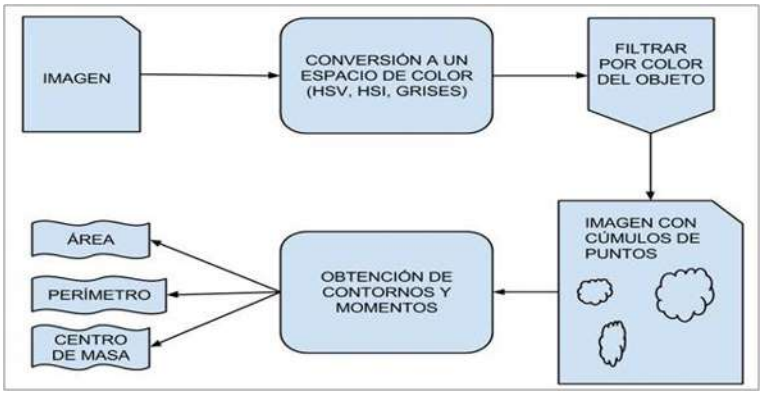

Figura 4.1: Proceso para la obtención del centro de masa de los objetos que se filtran por medio del color en una imagen, permitiendo la identificación y seguimiento de los mismos.

4.1. Etapa de captura

El sistema de visión computacional inicia con la captura del vídeo. Se deben considerar algunos aspectos importantes. El primer aspecto debe ser la iluminación, la cual debe ser homogénea para evitar brillos que dificulten ver los contornos del objeto, otro aspecto a considerar es la cámara, la cual es considerada como el elemento más importante del sistema y debe ser seleccionada de acuerdo a los parámetros de la aplicación, debemos considerar valores tales como la resolución, la velocidad de captura (cuadros por segundo o FPS por sus siglas en Inglés de *Frames per Second*), envío de datos a la computadora, entre otros.

La importancia del software utilizado, en este trabajo las bibliotecas OpenCV, recae además en el hecho de que facilita mucho el trabajo, es una API por sus siglas en Inglés de *Application Programming Interface* de aproximadamente 300 funciones que se caracterizan por su uso, ya que es libre tanto para su uso comercial como no comercial, además de que utiliza librerías numéricas externas, aunque puede hacer uso de alguna de ellas y un gran entorno para el cálculo numérico y simbólico y muchos más [DelaEscalera15]. Con la biblioteca de OpenCV se tiene la posibilidad de administrar una o más cámaras e ir capturando los frames del vídeo, en la imagen 4.2 se muestra como es que el robot ve

el escenario donde se encuentra un objeto de color, el cual es el que debe de identificar y realizar su seguimiento. Cabe mencionar que en este trabajo presentado se utiliza el lenguaje Python para la realixacion de las tareas mencionadas.

Figura 4.2: Escenario que ve el robot con la cámara, en el lado izquierdo se muestra una perspectiva más lejana del objeto, en el lado derecho el objeto tomado más de cerca.

La captura de imágenes se realiza en el espacio de color HSV, el código para hacer la captura de los frames requiere de una variable para guardar los datos de un frame y el dispositivo de captura, el fragmento de código como ejemplo que nos permite realizar la captura se muestra en el listado 4.1.

Listado 4.1: Líneas de código para la captura de vídeo en OpenCV.

```
cap = cv2.VideoCapture(0)
while(True):
# Capture frame-by-frame
    ret, frame = cap.read()
    frame = cv2.cvtColor (frame, cv2.COLOR_RGB2HSV)
```

4.2. Etapa de pre-procesamiento

El pre-procesamiento consiste en la aplicación de técnicas que permitan el realce o mejoramiento de algunas características importantes en las imágenes originales para facilitar el proceso de segmentación. En un sistema de visión computacional es necesario aplicar técnicas que conserven los contornos y homogeneicen la luz del ambiente ya que éste es un

factor muy importante para el sistema, la luz puede afectar mucho en las imágenes tomadas y con ellos causar errores.

La ventaja de trabajar con diferentes espacios de color es que de alguna manera se deben eliminar algunos brillos o reflejos de luz en la imagen y hacer posible saturar más los colores para que sea más fácil que el software detecte el color deseado, otra ventaja es que puede homogeneizar más la luz en toda la imagen y eso facilita mucho más el trabajar con el vídeo tomado para la etapa de procesamiento y filtrado.

Existen varios espacios de colores para la representación de imágenes, algunos de los más comunes son: BGR que representa al azul, verde y rojo, HSV que representa matiz, saturación y brillo y HSI que representa matiz, saturación y luminosidad.

En la figura 4.3 se muestra la comparación entre la imagen original tomada por la cámara en un espacio de color BGR y la imagen en un espacio de color HSV que se basa en saturación y brillo. Se puede observar que el espacio de color HSV resalta la saturación de los colores, dando mayor intensidad y colores más vivos. En OpenCV se logra con la línea de código mostrada en el listado 4.2.

Listado 4.2: Línea de código que muestra la función que permite cambiar de un espacio de color a otro.

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

Donde en la variable `hsv` se almacena el frame que se tomó en un espacio de color BGR ahora convertido a un espacio de color HSV

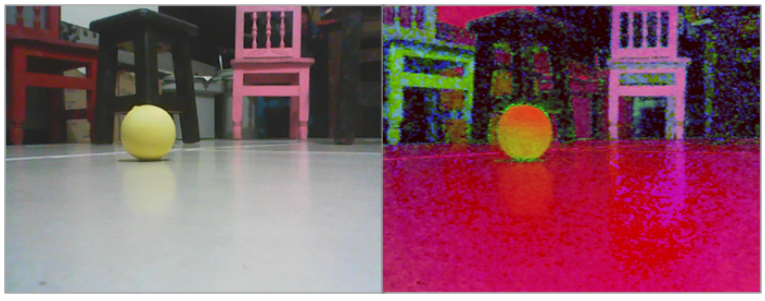

Figura 4.3: Imagen tomada una vez que se aplicó el filtro HSV

En la figura 4.4 se muestra la misma imagen, pero ahora en el espacio de color HSI

que se basa en el matiz, saturación y luminosidad de los colores, en la cual se ven menos contornos del objeto del color amarillo, por lo tanto sería más difícil acercarse al objeto y tomarlo ya que no se alcanza a detectar la mayor parte del objeto.

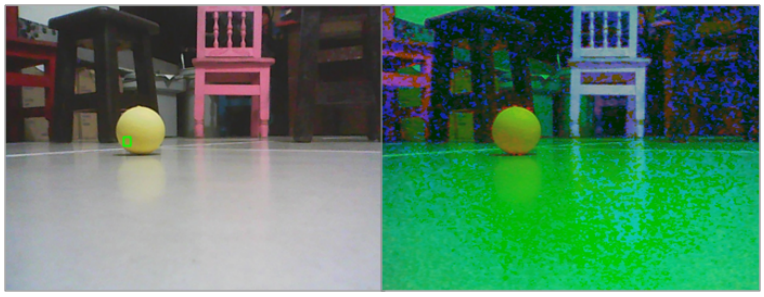

Figura 4.4: Imagen tomada una vez que se aplicó el filtro HSI

De la imágenes en la figura 4.3 y la figura 4.4 podemos concluir que algunos espacios de color, resaltan más la saturación de algunos colores, así como también algunos colores pueden verse invertidos, es decir no se muestra el color tal cual, se muestra el color opuesto (el negativo), en el caso del espacio de color HSV se muestra que el objeto a identificar (la pelota amarilla en la imagen) se percibe la mitad de un color rojo, esto se debe al contraste de la imagen y la saturación de color debido a la iluminación.

4.3. Filtrado del objeto mediante técnica de color

Una vez que se han establecido los valores en el espacio de colores para el objeto de interés, se procede a filtrar la imagen mediante la técnica de valor de umbral, obteniéndose cúmulos de píxeles en el rango especificado, que puede no identificar la forma del objeto, pero si la posición en donde se encuentra dentro de la imagen, ver la figura 4.5 para observar el ejemplo de filtrado en una imagen en el espacio de color HSV.

La técnica de umbralización consiste en determinar un cierto rango de valores máximo y mínimo para que el programa determine si el objeto capturado en los frames es del color deseado, este rango consta de dos vectores que contienen tres elementos cada uno, un vector que indica un rango mínimo es decir por abajo de color deseado y el otro que indica un rango máximo, es decir mayor al valor deseado, para dar una cierta tolerancia al

color, por si se viera afectado el color real, por alguna sombra o destellos de luz, nuestro sistema realiza dicha acción de filtrado por umbral con las líneas de código mostradas en el listado 4.3.

Listado 4.3: Para filtrar un color se especifica en dos arreglos el rango de los valores en el espacio de color utilizado, en el ejemplo el color amarillo.

```
lower_yellow = np.array([2, 70,190])
upper_yellow = np.array([45,105,240])
mask = cv2.inRange(hsv,lower_yellow, upper_yellow)
```

En la figura 4.5 se observa el cúmulo de píxeles en la imagen tomada por el robot con el espacio de colores HSV, en la imagen se muestra la región de color amarillo que alcanzó a percibir con el umbral utilizado.

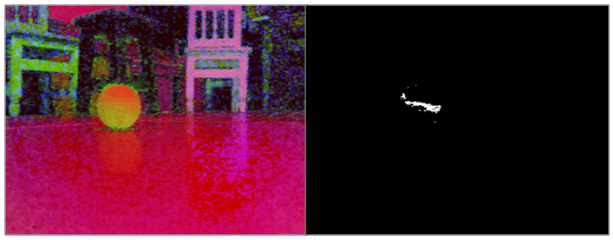

Figura 4.5: Una vez aplicado el espacio de color en la imagen con el filtro HSV.

En la figura 4.6 se muestra ahora el cúmulo de píxeles obtenidos en un filtrado de umbral con el espacio de color HSI el cual nos muestra la imagen correspondiente al color amarillo del objeto que alcanzó a percibir basándose en la luminosidad del ambiente.

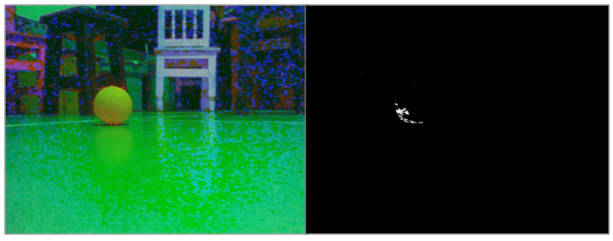

Figura 4.6: Ejemplo del filtrado por umbral: una vez transformada la imagen al espacio de color HSI (lado izquierdo), píxeles resultantes del filtrado por color (lado derecho).

Y por último en la figura 4.7 se muestra el cúmulo de píxeles obtenido en el espacio

de color BGR, el cual muestra la imagen tal y como ésta fue obtenida por el dispositivo de captura, este espacio de color no altera la imagen ya que es en la forma en que generalmente se perciben los colores, en la combinación de los colores, rojo, azul y verde, cabe mencionar que este espacio de color no se basa en saturaciones ni brillos, además de no eliminar destellos ni homogeneizar la luz del ambiente.

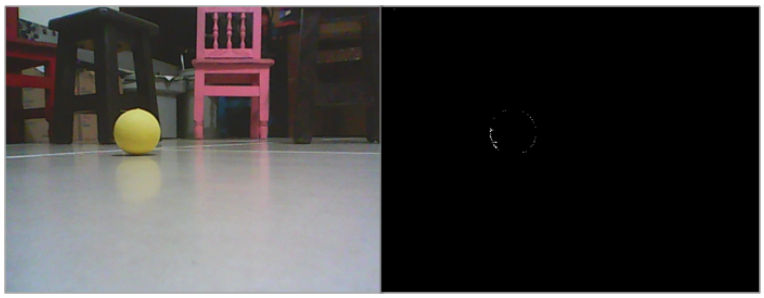

Figura 4.7: Filtro por umbral utilizando el espacio de color BGR.

4.4. Segmentación

La segmentación de la imagen nos permite separar la información que nos interesa de la que no es de utilidad para lograr los objetivos. Existen diferentes métodos dentro del procesamiento de imágenes, uno de ellos es definir una región de interés (ROI por sus siglas en Inglés de Region of Interest) dentro de la cual se definirá un área dentro de la imagen para analizar los datos contenidos dentro de ella.

Con la información de cúmulos de píxeles, se hace posible la obtención de los contornos, de los cuales se puede recabar información característica, como el área, perímetro, centro de masa, puntos del cuadrado que encierra el contorno, por mencionar algunos. El centro de masa del contorno mayor en los cúmulos de píxeles nos indica la posición del objeto de interés. En la figura 4.8 se observa como el robot visualiza los la imagen original, después como se observa la imagen en tono de grises, como se obtienen los cúmulos y como se delimita el contorno de la imagen, estos resultados se obtuvieron con los parámetros de umbral que se muestra en el listado 4.4.

Listado 4.4: Reduciendo el umbral de detección para el color azul.

```
lower_blue = np.array([60,70,130])
upper_blue = np.array([125,155,225])
```

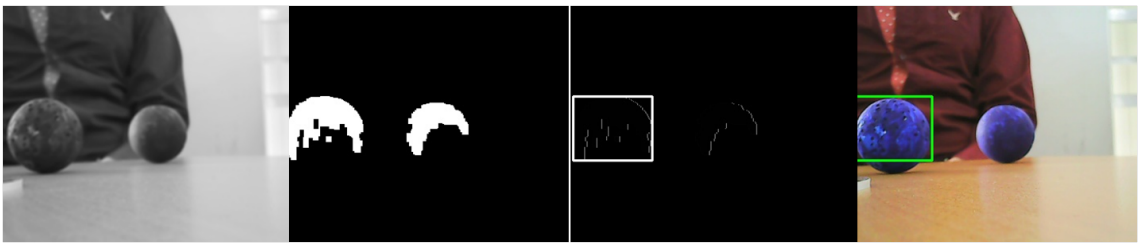

Figura 4.8: Proceso de filtrado de contornos con un umbral reducido.

Con los valores reducidos del umbral, los cúmulos de píxeles que detecta el robot son menos, debido a que el campo de visión se reduce en colores, en comparación con la figura 4.5 en este caso se alcanza a percibir mayor cúmulo y por lo tanto mayor área del objeto debido a que tiene un umbral mayor. En la figura 4.9 se observa que las zonas que se detectan del objeto incrementan, aumentando el umbral.

Listado 4.5: Umbral de detección para el color azul.

```
lower_blue = np.array([30,40,100])
upper_blue = np.array([155,185,255])
```

Figura 4.9: Proceso de filtrado de contornos con un umbral mayor.

Y por último en el listado 4.4 se muestra el umbral más alto que alcanzó a detectar sin perder el objeto, en la imagen 4.10 se muestran los contornos que alcanzo a percibir con dichos parámetros, se observa que detecta casi por completo el objeto de interés.

Listado 4.6: Umbral de mayor detección para el color azul.

```
lower_blue = np.array([30,40,90])
upper_blue = np.array([165,195,245])
```

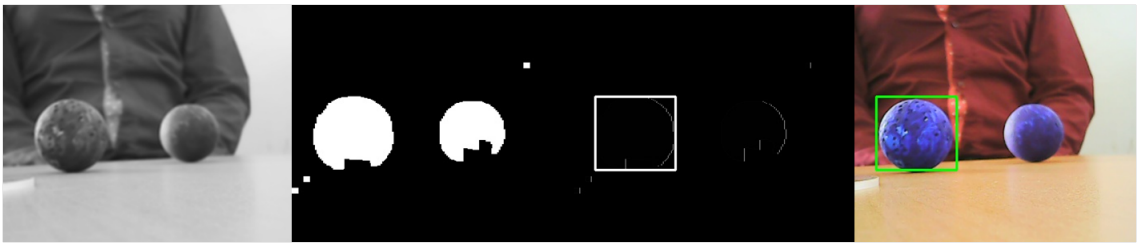

Figura 4.10: Proceso de filtrado de contornos con el máximo umbral.

El código para realizar la segmentación mostrada en las imágenes es el del listado 4.7

Listado 4.7: Código para realizar la segmentación.

```
contornosI,jerarquia =
    if len(contornosI) > 0:
        i_max=EncuentraContornoMayor(contornosI)

def EncuentraContornoMayor (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  # regresa el contorno con mayor per metro
```

4.5. Seguimiento de objetos mediante Visión Computacional

Los robots usualmente navegan autónomamente en ambientes dinámicos, de tal forma que necesitan detectar y evadir objetos. Cuando se utilizan cámaras, la detección de obstáculos puede realizarse a través de la reconstrucción 3D. El método de visión estéreo (uso de dos cámaras) es la forma clásica para la obtención de información 3D, se basa en determinar la correspondencia de los píxeles entre dos cámaras y realizar la triangulación, el inconveniente es que este método falla con superficies no texturizadas. Algunos trabajos se han realizado en la navegación de robots basados en visión y control logrando generar el ambiente del robot sin una reconstrucción explicita del ambiente en 3D.

La representación visual de objetos de interés alrededor del robot que conlleva la identificación de dichos objetos en el actual campo de visión, puede mejorar el control en el ambiente del robot al tener mayor información para la toma de decisiones.

Usualmente es necesario que rápidamente el robot explore nuevas áreas del ambiente para observar objetos conocidos y entonces actualizar su posición o realizar una tarea sobre el objeto (como recogerlo). Una forma de detectar objetos de interés es mediante el color, para ello se hace necesario trabajar en un espacio de color que nos permita controlar características de iluminación.

El robot para seguir el objeto debe de tener ya la imagen segmentada e identificar en que coordenadas se encuentra, lo que hace el robot es identificar en que lado de la pantalla se encuentra el objeto de interés, si es del lado derecho se encontrará en una coordenada positiva en x, si el objeto de interés se encuentra del lado izquierdo de la pantalla se encontrará en una coordenada negativa del eje x, tomando como referencia la parte central de la imagen, para saber que tan cerca o lejos se encuentra del objeto se utiliza el sensor infrarrojo que determina la distancia entre el robot y dicho objeto, cuando el robot se encuentra a una distancia determinada que se considere lo suficientemente cerca para con la longitud de sus pinzas poder tomarlo, el robot verificara que el sensor mande la señal de proximidad y la cámara detecte el objeto, si ambas condiciones se cumplen, el robot mandará la señal al servomotor para poder atrapar la esfera. En la figura 4.11 se muestra un diagrama de flujo del algoritmo de seguimiento del robot.

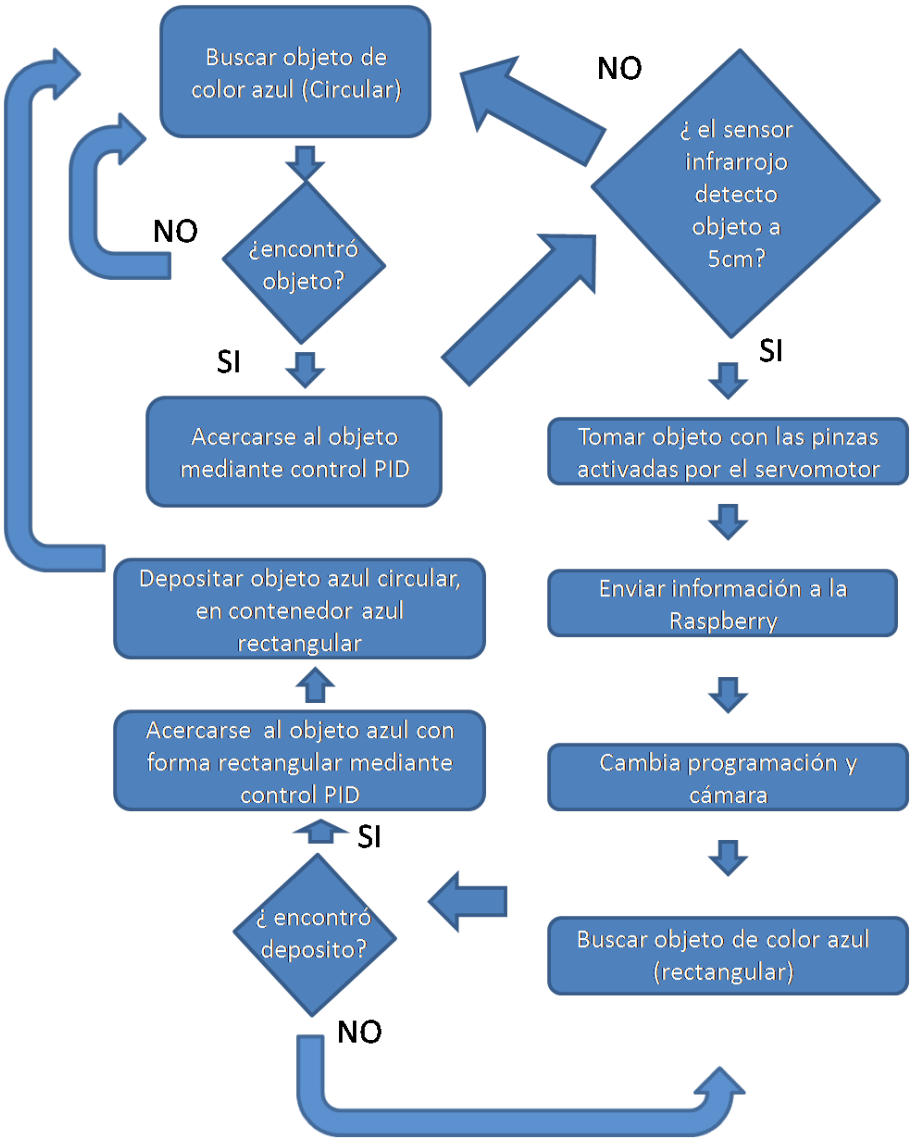

Figura 4.11: Diagrama de flujo del algoritmo del robot de servicio, para el seguimiento del objeto.

En la figura 4.12 se muestra un frame de cómo el robot percibe los objetos en su ambiente, después de pasar por las etapas pre-procesamiento y segmentación, identificó el objeto de interés, indicado por un rectángulo de color en la imagen en el lado izquierdo, el tamaño del rectángulo lo establece el contorno de mayor área encontrado en la etapa de segmentación, lo que da la sensación de que se identifica solamente una pequeña parte de dicho objeto. Para que el robot pueda alinearse con el objeto se utiliza el control PID, el

cual hará los movimientos necesarios del robot para fijar en el centro de la imagen el objeto de interés.

Figura 4.12: Contornos que el robot distinguió del objeto.

En la figura 4.12 se percibe que aunque todo el objeto es de color amarillo, el robot no alcanza a percibir el contorno de todo el objeto, debido a que el umbral de identificación no es tan amplio, esto reduce la visión del robot, ya que son menos los cúmulos de píxeles que alcanza a distinguir y sólo distingue la zona de la esquina inferior derecha del objeto. En la figura 4.13 muestra el mismo objeto pero una vez que el robot se alineó un poco más a él, en esta imagen el robot alcanza a percibir un poco más de área del objeto, es decir el contorno que distingue es mayor.

Figura 4.13: Contornos que el robot distinguió del objeto en una perspectiva más cercana.

Si el robot se encuentra perpendicular al objeto, los filtros eliminan más el ruido causado por los brillos en la imagen y el robot puede percibir mayor área del objeto y con eso es más fácil seguir dicho objeto. Ver imagen 4.15

Una vez que el robot está centrado y de frente al objeto y los filtros de color

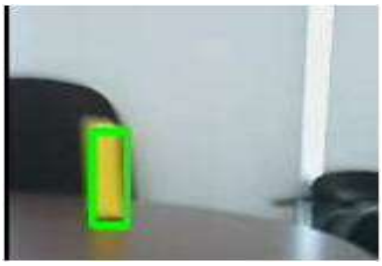

Figura 4.14: Contornos que el robot distinguió encontrando mayor área en el objeto.

eliminan la mayor cantidad de brillos que pudieran meter ruido en los frames, es ahí cuando el error del control PID se reduce a 0 y el robot sólo tiene que avanzar en forma lineal para atrapar el objeto ya que éste se encuentra en la coordenada 0 del eje x de la pantalla, es decir está totalmente perpendicular al objeto como se muestra en la figura 4.15.

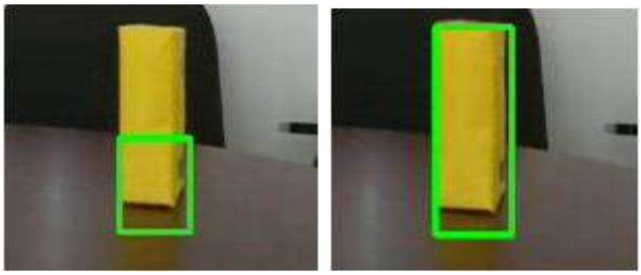

Figura 4.15: Contornos con mayor área que el robot captó del objeto.

Y por último una vez que el robot se encuentra de frente al objeto y muy cerca, es decir cuando ocupa la mayor parte de la pantalla, es cuando el robot enciende el sensor de distancia y si el objeto se encuentra dentro del área de alcance del robot, cierra las pinzas mediante el servomotor para poder tomar el objeto.

4.6. Control de desplazamiento del robot móvil

La rama de sistemas de control es de suma importancia para todo ingeniero, su utilización ha sido vital en el avance de la ingeniería y la ciencia, permitiendo liberar al hombre de tareas repetitivas y su exposición a ambientes inseguros que pueden ser ejecutadas fácilmente por algún sistema de control automático. Al estudiar la ingeniería de control se

deben definir la terminología básica para describir los sistemas de control. En particular, un sistema de control se puede considerar como una caja negra con una entrada y una salida [Lázaro08]. En la figura 4.16 vemos un diagrama general del sistema de control.

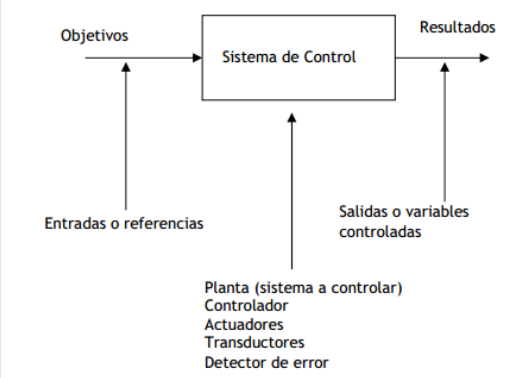

Figura 4.16: Diagrama general de un sistema de control.

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 [Haj-Ali04], por lo que lo convierte en la técnica de control más común. Se basa en minimizar el error que se obtiene de la diferencia de 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 [Almeida98], en el control de robots auto-balanceados en dos ruedas [Sun10], además de los seguidores de objetos por mencionar algunas aplicaciones.

Control PID: El controlador PID (Proporcional, Integral y Derivativo) es un controlador realimentado cuyo propósito es hacer que el error en estado estacionario entre la señal de referencia y la señal de salida de la planta, sea cero de manera asintótica en el tiempo, lo que se logra mediante el uso de la acción integral. Además el controlador tiene la capacidad de anticipar el futuro a través de la acción derivativa que tiene un efecto predictivo sobre la salida del proceso [Moreno01]. La acción que proporciona cada fase del control

es la siguiente:

- Control proporcional: su principal acción consiste en medir la referencia y compararla con el valor actual tomado por el sensor y con ello determinar el error.
- Control derivativo: consiste en comparar el error anterior con el error actual.
- Control integral: consiste en la sumatoria de todos los errores anteriores.

En la figura 4.17 se muestra el diagrama de control utilizado en el robot para lograr el seguimiento del objeto y poder acercarse a él.

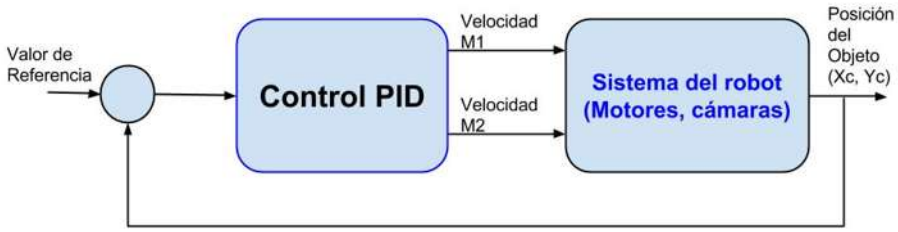

Figura 4.17: Diagrma de control del robot.

4.6.1. Señales PWM

En nuestro sistema se implementó el control PID el cual manda señales a los motores para poder encenderlos o apagarlos, ésto se tuvo que hacer mandando la información de la Raspberry Pi al microcontrolador Arduino, ya que la Raspberry no es capaz de mandar señales PWM, con este tipo de señales podemos convertir una acción digital de los motores en una señal analógica, es decir en vez de tener sólo 2 valores, el de encendido y apagado para realizar el seguimiento del objeto, se puede reducir o aumentar la velocidad, en una o ambas llantas, así se logra un seguimiento más eficiente, ya que no sólo tiene estados de on/off sino puede ir variando la velocidad de los motores para hacer la acción de seguimiento. La modulación por ancho de pulsos (también conocida como PWM, por siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

Los sistemas analógicos, tales como fuentes de alimentación lineales, tienden a generar una gran cantidad de calor, ya que son básicamente resistencias variables que llevan una gran cantidad de corriente. Los sistemas digitales por lo general no generan tanto calor, casi todo el calor generado por un dispositivo de conmutación es durante la transición (que se realiza rápidamente), esto es porque la energía sigue la ecuación 4.1.

$$P = VI \tag{4.1}$$

Donde P es la potencia e watts, I es la correinte en amperes y V es el voltaje.

Si el voltaje o la corriente es cercano a cero entonces la potencia será cercana a 0, PWM saca el máximo partido de este hecho.

Las señales PWM utilizan un valor llamado ciclo de trabajo que es el porcentaje del ciclo de trabajo realizado por el sistema, en otras palabras el valor del ciclo de trabajo define cuantó porcentaje de voltaje dejará pasar al sistema, es decir si es 0, dejará pasar 0 volts, si el ciclo de trabajo es de 100 dejará pasar todo el voltaje al sistema. En la figura 4.18 se observa la variación en porcentaje de diferentes ciclos de trabajo.

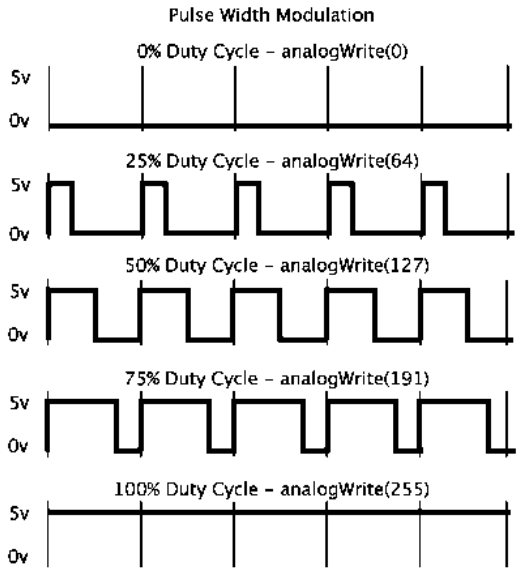

Figura 4.18: Ciclo de trabajo de un sistema con PWM.

Así el ciclo de trabajo puede variar el ciclo de trabajo y el voltaje del sistema, para evitar tener un control on/off, de encendido y apagado de motores, gracias a las

señales PWM, se puede reducir la velocidad en los motores del robot y así evitar que se apaguen inmediatamente, sino que mas bien vayan reduciendo su velocidad cuando tengan que hacerlo. en la figura 4.19 se muestra el ciclo completo de trabajo y el valor activo del parámetro duty [Kuphaldt15]. .

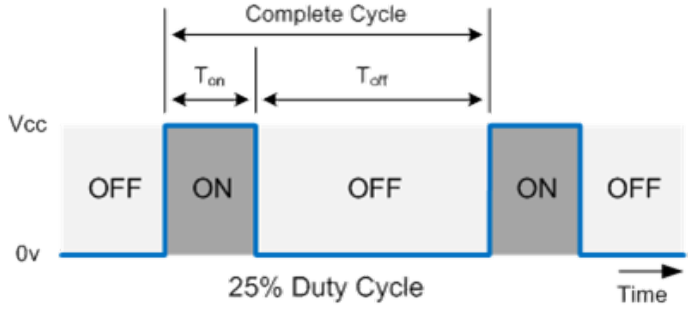

Figura 4.19: Ciclo de trabajo completo.

En nuestro caso las señales PWM las utilizamos para con ellas controlar la velocidad de los motores del robot, es decir en vez de mandar sólo señales para prender y apagar los motores, podemos moderar el ciclo de trabajo, mejor dicho el voltaje que alimenta a los motores y con ello variar la velocidad, ya sea para mantenerla constante, reducir la velocidad o aumentarla.

Capítulo 5

Experimentación y Resultados

El objetivo del robot móvil es visualizar su ambiente por medio de las cámaras, dicho robot se encontrará en un escenario de rodeado por tablas de madera, después de eso identificar el objeto de interés, ya que lo identificó, acercarse a él mediante control PID y tomar el objeto.

Se realizaron 3 experimentos del robot móvil autónomo, el primero sirve para identificar el alcance de visión del robot a diferentes escenarios de iluminación. El segundo experimento define la velocidad de búsqueda y seguimiento del objeto identificado a diferentes escenarios. Finalmente el tercer experimento se refiere a las veces que con éxito se realizó la tarea para la cual se programó el robot móvil.

5.1. Experimento 1: Medir el rango de visión del robot.

El primer experimento que se realizó, fue plantear varios escenarios, en los cuales, el robot tiene que identificar una esfera de unicel del número 4, pintada de color azul, y analizar la máxima distancia a la cual logra identificarla el robot, la experimentación se repitió 5 veces para cada uno de los escenarios siguientes.

5.1.1. Escenario 1: Luz ambiente al costado izquierdo del robot.

Colocando el robot de frente al objeto con la luz ambiente a un costado, en nuestro caso la mayor fuente de luz observada se encontraba del lado izquierdo, la máxima distancia

a la que logró identificar el objeto fue de 300 cm, como se muestra en la figura 5.1 esta fue la mayor distancia de identificación de un objeto bajo las condiciones controladas de iluminación.

Figura 5.1: Distancia de identificación con la luz ambiente en el costado izquierdo.

5.1.2. Escenario 2: Luz ambiente a contraluz del robot.

Colocando el robot de frente al objeto con la luz ambiente a espaldas, la máxima distancia en que se logró percibir la esfera de unicel fue de 220 cm aproximadamente, en la figura 5.2 se muestra las posición del robot y el objeto a identificar en las condiciones de iluminación del experimento.

Figura 5.2: Distancia de identificación del objeto a contraluz del robot.

5.1.3. Escenario 3: Luz ambiente de frente al robot.

Colocando el robot de frente al objeto con la luz ambiente de frente, el sistema de visión en el robot no logró identificar el objeto, esto se debió a que la luz provoca un cambio de color en el objeto (obscuro) y la cámara no identifica el color real del objeto, se observa un objeto totalmente negro y ese valor no se encuentra dentro del umbral del color azul que el robot tenía establecido para la identificación de objetos, se sugiere la identificación de formas geométricas para sobrellevar esta problemática, ver la figura 5.3 que ilustra la posición del robot y el objeto en este experimento.

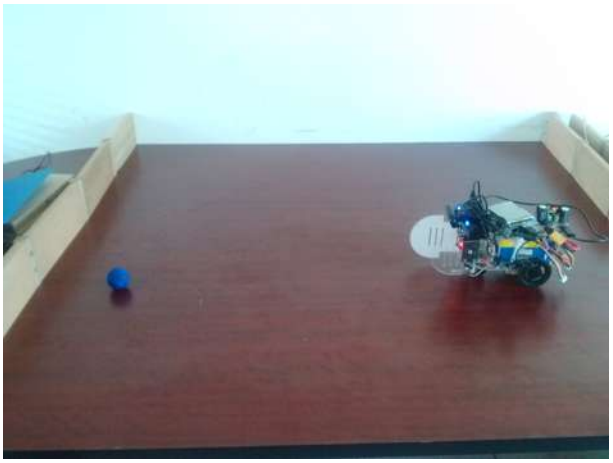

Figura 5.3: Experimento con la luz ambiente de frente al robot para obtener la distancia a la cual se identifica el objeto.

Finalmente, en la figura 5.4 se muestra el escenario bajo el cual se realizaron los experimentos anteriores, describiendo que fue un ambiente semi-controlado, con iluminación por ambos costados como se muestra en las imágenes y una cortina que nos permite controlar la cantidad de luz en el espacio de trabajo. La finalidad de describir el escenario de experimentación, se debe a que los sistemas de visión computacional son muy sensibles a la iluminación en que se desempeñan.

Figura 5.4: Escenario de experimentación.

5.2. Experimento 2: Medir la velocidad de desplazamiento del robot.

El segundo experimento que se realizó, fue plantear varios escenarios, en los cuales, el robot tenía que desplazarse hacia el objeto de interés desde varios ángulos, la distancia que el robot tenía que recorrer para llegar al objeto fue de 50 cm, se realizaron 5 pruebas para cada caso y se promedió el tiempo que le tomó al robot realizar la tarea programada, en dicha prueba sólo se probó el tiempo que el robot duraba en acercarse al objeto, sin capturarlo, sólo se tomó el tiempo de desplazamiento desde donde se encontraba el robot, hasta la posición del objeto.

5.2.1. Ambiente 1: robot de frente al objeto.

Colocando el robot de frente al objeto, es decir a 0° con respecto del objeto, como se muestra en la figura 5.5, el tiempo promedio en llegar al objeto fue de 3.94 segundos. En la tabla 5.1 se muestran los valores obtenidos.

Prueba	Tiempo
1	3.3 s
2	5.1 s
3	4.1 s
4	3.7 s
5	3.5 s

Tabla 5.1: Resultados de la experimentación con el robot a 0° del objeto.

Figura 5.5: Al encontrarse el robot a 0° del objeto, la cámara del robot queda de frente al objeto a identificar, la tarea principal es seguir la trayectoria al objeto correctamente.

5.2.2. Ambiente 2: robot desplazado a la izquierda del objeto 45°.

Colocando el robot de frente al objeto a 45° en relación al objeto del lado izquierdo, como se muestra en la figura 5.6 el tiempo promedio en llegar al objeto fue de 17.9 segundos. La tabla 5.2 muestra los resultados en relación al tiempo que al robot le tomó llegar al objeto, la búsqueda del objeto se hace en el sentido de giro contrario a las manecillas del reloj.

Prueba	Tiempo
1	15.3 s
2	18.2 s
3	20.9 s
4	18.1 s
5	17.5 s

Tabla 5.2: Resultados de la experimentación con el robot a 45° del objeto desplazado sobre el lado izquierdo del robot.

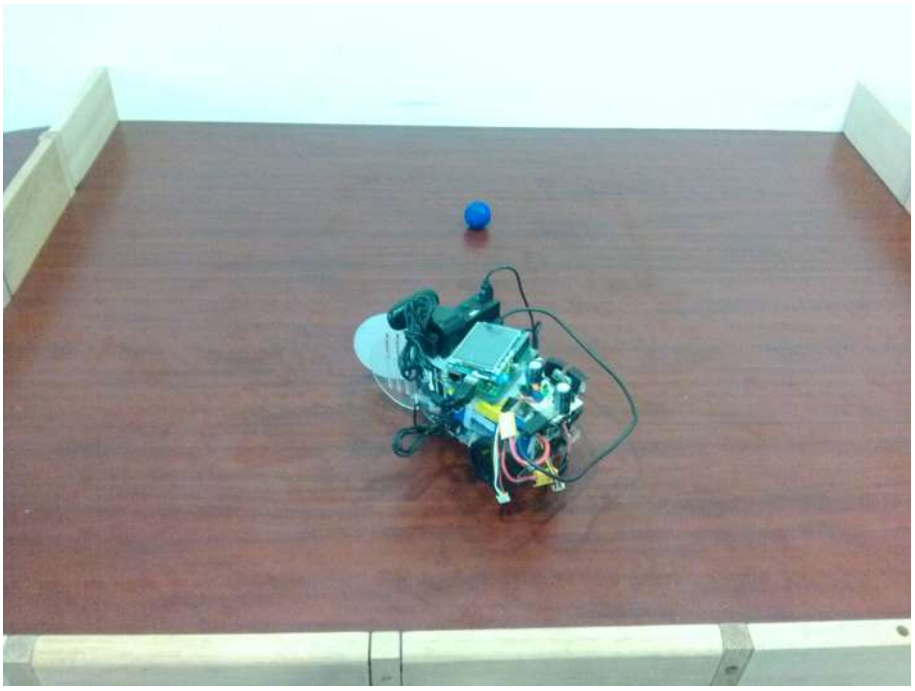

Figura 5.6: La cámara del robot observa al costado izquierdo del objeto, al no identificar el objeto en la escena se hace la búsqueda.

5.2.3. Ambiente 3: objeto a espaldas del robot, giro de 180°.

Colocando el objeto a espaldas del robot, es decir quedando en un ángulo de 180° en relación al objeto, como se muestra en la figura 5.7, el tiempo promedio en realizar la búsqueda y llegar al objeto fue de 11.8 segundos. En la tabla 5.3 se muestran los tiempos obtenidos 5 pruebas realizadas para este experimento.

Prueba	Tiempo
1	10.4 s
2	13.2 s
3	12.5 s
4	12.7 s
5	10.5 s

Tabla 5.3: Tiempos obtenidos para capturar el objeto, con el robot a 180° del objeto.

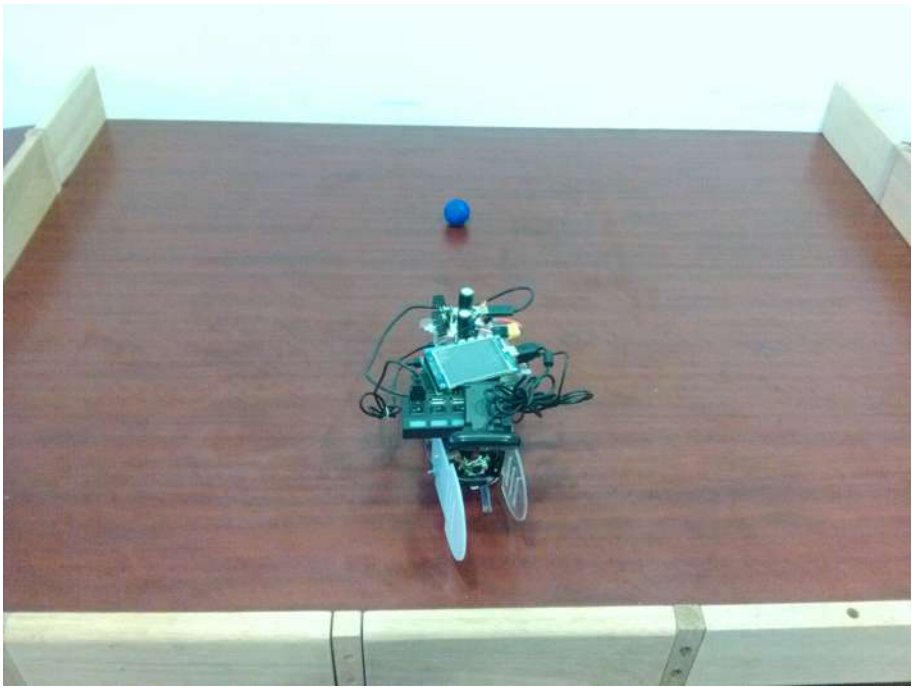

Figura 5.7: En la imagen se muestra el objeto a espaldas del robot, que debe buscar y capturar.

5.2.4. Ambiente 4: robot desplazado a la derecha del objeto, 315°.

Colocando el robot de frente al objeto a 315° en relación al objeto, como se muestra en la figura 5.8, el tiempo promedio en realizar la búsqueda y captura del objeto fue de 9.0 segundos. En la tabla 5.4 se muestran los resultados de tiempos obtenidos de la experimentación.

Prueba	Tiempo
1	8.8 s
2	10.0 s
3	9.4 s
4	9.7 s
5	7.5 s

Tabla 5.4: Resultados de tiempos obtenidos en la experimentación con el robot a 315° del objeto.

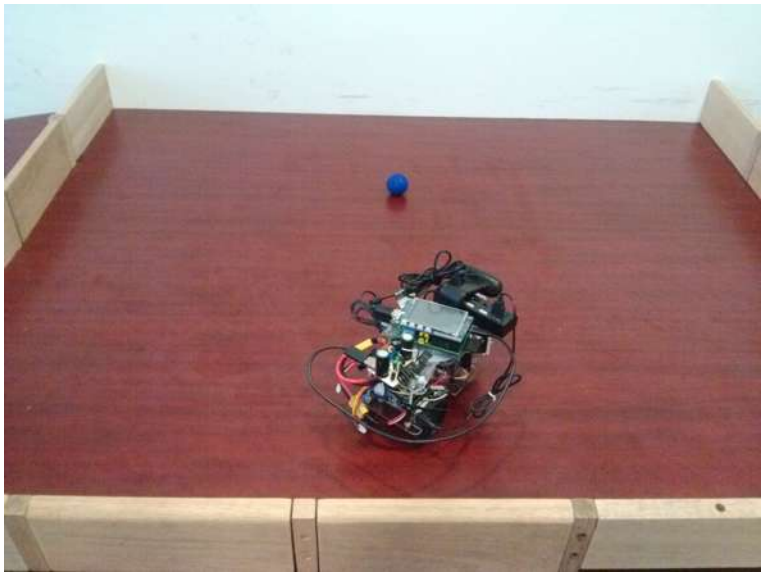

Figura 5.8: Robot observando al costado derecho de su eje, en una posición a 315° del objeto.

5.3. Experimento 3: Efectividad del robot para capturar el objeto.

El tercer experimento que se realizó, consistió en medir el número de veces que el robot buscó y capturó el objeto en cuestión. El ambiente para dicho experimento se describe como un rectángulo rodeado de paredes de 10 cm. de alto y 2.5 cm de espesor, el rectángulo tiene las dimensiones 140 x 110 cm ancho x alto respectivamente, la figura 5.9 nos muestra las condiciones de iluminación y forma del ambiente experimental.

Figura 5.9: Ambiente donde se realizaron las pruebas.

Para una mayor descripción de las condiciones de iluminación en el ambiente observar las imágenes en la figura 5.10, el experimento se realizó en un cuarto iluminado con luz natural, luz que entra por las ventanas que se encuentran en los costados de dicho cuarto.

Figura 5.10: Iluminación del ambiente donde se realizaron las pruebas.

La distancia de pruebas fue de 50 cm. de distancia entre el objeto y el robot en todos los experimentos, en línea recta, con la mayor iluminación en el ambiente al costado derecho del robot, como se muestra en la imagen 5.11.

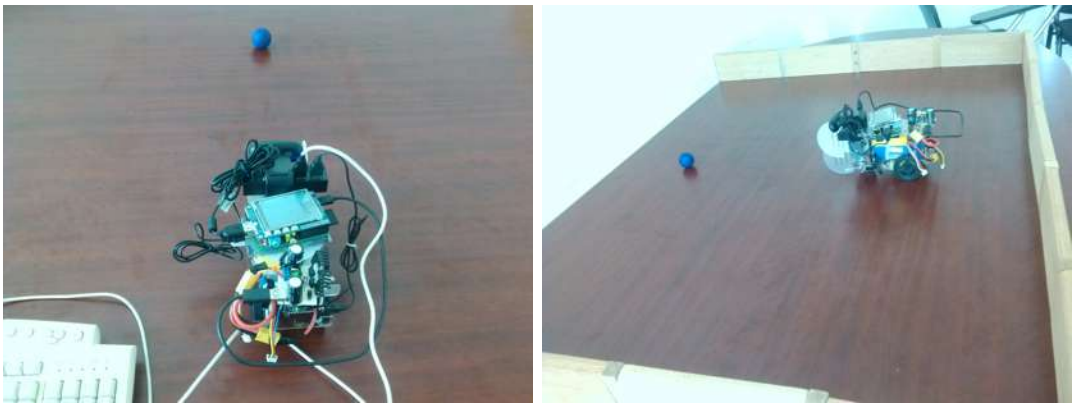

Figura 5.11: Posición entre el objeto y el robot a una distancia de 50 cm.

La tarea del robot es capturar el objeto a la distancia de 50 cm en línea recta, este experimento se realizó 25 veces. La efectividad con que se realiza la tarea se establece por

la ecuación 5.1. La tabla 5.5 nos muestra los resultados obtenidos en este experimento.

$$EF = \frac{NVE}{NVT} * 100 \tag{5.1}$$

En donde EF es la efectividad en porcentaje; NVE se refiere al número de veces que efectivamente se realizó la tarea por el robot; y NVT es el total de intentos realizados en la experimentación.

Prueba	Efectividad	Tiempo
1	1	16.1 s
2	1	15.5 s
3	0	12.0 s
4	1	18.0 s
5	1	15.3 s
6	0	14.8 s
7	1	17.5 s
8	0	15.0 s
9	1	12.1 s
10	1	15.5 s
11	1	13.2 s
12	1	12.8 s
13	0	15.2 s
14	0	13.8 s
15	1	13.4 s
16	1	15.5 s
17	0	18.2 s
18	1	15.4 s
19	1	15.5 s
20	0	18.0s s
21	1	17.7 s
22	1	18.0 s
23	1	11.3 s
24	1	14.1 s
25	1	14.9 s

Tabla 5.5: Tabla de tiempos y cumplimiento con la tarea por el robot.

La efectividad del robot es

$$EF = \frac{18}{25} = 0.72 = 72\,\% \tag{5.2}$$

en la tabla se colocó un 1 en los casos que el robot capturó el objeto y el tiempo que tardó

en realizar la actividad y un 0 en caso contrario, en dichos casos el robot tomó efectivamente el objeto con un tiempo promedio de 15.1 s a una iluminación natural dentro de un cuarto.

Capítulo 6

Conclusiones y Trabajos Futuros

6.1. Conclusiones

Como conclusiones del presente trabajo se señalan las siguientes:

- Se implementó un prototipo de un robot móvil autónomo. En esta tesis se describen las partes de hardware y software que permiten realizar la tarea para la cual se diseñó el prototipo.

- El control PID resultó eficiente para el desplazamiento del robot, al seguir la trayectoria que el sensor (cámara) en conjunto con el software de procesamiento.

- El tiempo de desplazamiento promedio en la experimentación a una distancia del objeto de 50 cm. al robot, una vez que se visualizaba un objeto de interés es de 3.94 segundos, resultando una velocidad de desplazamiento del robot de

$$\frac{50cm}{3.94s} = 12.69\frac{cm}{s}$$

- El tiempo promedio que le toma al robot en realizar la tarea de buscar y tomar el objeto es de 15.1 segundos, ésto incluye el desplazamiento y la acción de tomar el objeto, observándose una eficiencia del 72 %.

- El desempeño del robot depende mucho del hardware y de las características de procesamiento, la cantidad de imágenes que deben procesarse para determinar el dato que

permite el desplazamiento, lleva a recomendar un procesador con mayor capacidad.

- La aplicación directa que se puede tener de este tipo de robots móviles autónomos es como robots de servicio. Una de sus tareas es identificar ciertos objetos en situaciones de riesgo, por ejemplo detectar un extintor y poderlo tomar para el caso de un incendio; también como robot de servicios domésticos, para facilitar las labores del hogar o de alguna empresa o fábrica, realizando tareas repetitivas de búsqueda e identificación de ciertos objetos de interés para dicha dependencia, desarrollo que se deja como trabajos futuros.

- Se presenta el desarrollo de un sistema de visión computacional para la identificación de objetos. El sistema de visión implementado es aplicado a un robot móvil que identifica objetos de interés basándose en su color y forma y se desplaza hacia ellos. Se realizaron pruebas para identificar los colores azul y amarillo.

- La identificación de objetos por color permite que se obtengan las siluetas de los objetos, concluyendo que podemos tener objetos de diferentes formas y tamaños.

- El costo promedio del robot oscila en los 6000.00 pesos M.N., aún cuando la unidad de procesamiento no excede los $35 dólares, concluyendo en que es proyecto factible para instituciones de educación superior que cuentan con carreras que involucran la robótica, electrónica o computación.

- La luz es un factor que determina la eficiencia de los sistemas de visión y por ende el trabajo o tarea a desarrollar el robot.

6.2. Trabajos futuros

En trabajos futuros se plantean los siguientes:

- Implementar un robot de mayores dimensiones físicas, para poder manipular algún tipo de carga.

- Utilizar una microcomputadora de mayor poder de procesamiento, se propone el uso de la Raspberry Pi 2, que actualmente cuenta ahora con un procesador de cuatro núcleos;

es de considerarse también los sistemas embebidos Odroid que manejan procesadores con mayores ventajas de procesamiento.

- Implementar un dispositivo que capture un objeto (brazo del robot), de tal forma que pueda levantarlo de la superficie donde se encuentre al menos 2 centímetros, esto podrá permitir el ingreso a competencias en donde esa tarea es uno de los requerimientos, además de tener un prototipo que pueda ser utilizado en materias del área de electrónica, control y programación de computadoras.

- También se espera aumentar la visión del robot con cámaras de profundidad como el Kinect, que en un robot de servicio proporcionaría otras ventajas.

- Identificar formas geométricas en los objetos, con la finalidad de clasificar los objetos.

- Implementar el sistema de visión sobre un lenguaje de programación que no sea interpretado, como C o C++.

- Considerar el uso de programación concurrente sobre un sistema de procesamiento de varios núcleos.

Apéndice A

Código fuente del sistema de Visión

El código del sistema de visión computacional que permite la identificación de objetos y la comunicación con el sistema de control a través del puerto serial se lista a continuación.

```
import numpy as np
import cv2,cv
import math
import serial
import struct

def EncuentraContornoMayor (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  # regresa el contorno con mayor perimetro que existe en varC
```

```
cap = cv2.VideoCapture(1)
cap.set(3,480)
cap.set(4,320)
cols = cap.get(3)
rows = cap.get(4)
Xo=cols/2
Yo=rows/2
print rows, cols, Xo, Yo
connected = False
ser = serial.Serial('/dev/ttyUSB0',9600)

kernel = np.ones((5,5),np.uint8)

imagenBlanco = np.zeros((rows,cols/3,3),np.uint8)
imagenBlanco[:,:]=(255,255,255)
led=1
while(cap.isOpened()):
    ret, frame = cap.read() # capturar cada frame

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    grises = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    grises = cv2.GaussianBlur(grises,(5,5),1.5)
    lower_blue = np.array([30,40,90])
    upper_blue = np.array([165,195,245])
    mask = cv2.inRange(hsv, lower_blue, upper_blue)
    mask=cv2.dilate(mask,None,iterations = 5)
    mask=cv2.erode(mask,None,iterations = 2)

    cv2.imshow('grises',grises) # Mostrar el frame capturado
    cv2.imshow('erosionado/dilatado',mask) # Mostrar el frame capturado
    cv2.imwrite("imagen_cumulos-0.jpg",mask)

    contornosI,jerarquia = cv2.findContours(mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

```
if len(contornosI) > 0:
    i_max=EncuentraContornoMayor(contornosI)
    x,y,w,h = cv2.boundingRect(i_max)
    cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
    cv2.rectangle(mask,(x,y),(x+w,y+h),(255,255,255),2)
    M = cv2.moments(i_max)
    cx = int(M['m10']/M['m00'])-Xo
    cy = int(M['m01']/M['m00'])
    if w < h:
            radio=h
    else:
            radio=w
    '''
    if radio < cols/2:
            if mask != None:
                    circles=cv2.HoughCircles(mask,cv.CV_HOUGH_GRADIENT,2,50
            if circles != None :
                    c=circles[0][0]
                    cv2.circle(mask,(c[0],c[1]),c[2],(255,255,255),2)
                    cv2.circle(frame,(c[0],c[1]),c[2],(0,255,255),2)
    print "xc=%d yc=%d"%(cx,cy)
    '''
    signo='\n'
    ser.write(signo)
    ser.write(str(int(cx)).encode())
    led=ser.read()
    if led != 1:
            break
cv2.imshow('HSV',mask) # Mostrar el frame capturado
cv2.imshow('Frame Original',frame) # Mostrar el frame capturado
if cv2.waitKey(1) & 0xFF == ord('q'):
    cv2.imwrite("objetoidentificado-0.jpg",frame)
    cv2.imwrite("imagen_grises-0.jpg",grises)
```

```
        cv2.imwrite("imagen_contornos-0.jpg",mask)
        break

cap.release()
cv2.destroyAllWindows()
```

Apéndice B

Código fuente para el control del desplazamiento del robot

Código fuente en el lenguaje de programación del sistema arduino, que implementa el sistema de control PID que dirija al robot hacia el objeto.

```
#include "math.h"
#include<Servo.h>
Servo servo1;
#define dirM1_1 3
#define dirM1_2 5
#define dirM2_1 6
#define dirM2_2 9

int distanciaXcYc;
float error=0, error_anterior, proporcional, derivativo, integral;
float Kp=(float)1/850, Ki=0.000, Kd=(float)5/3;
int vel_motor1 = 90;
int vel_motor2 = 90;
int vel_base = 75;
int vel_apagado =0;
int referencia = 0;
void setup() {
```

```
 pinMode(A0, INPUT);
 servo1.attach(11);
 pinMode(dirM1_1, OUTPUT);
 pinMode(dirM1_2, OUTPUT);
 pinMode(dirM2_1, OUTPUT);
 pinMode(dirM2_2, OUTPUT);
 Serial.begin(9600);
 servo1.write(110);
 pinMode(A1, OUTPUT);
}
int leerSensores(){
  char signo;
  while(!Serial);
  distanciaXcYc = Serial.parseInt();

    if(distanciaXcYc == 0)
  {
          analogWrite(dirM1_1, LOW);
          digitalWrite(dirM1_2,LOW);
          analogWrite(dirM2_1, LOW);
          digitalWrite(dirM2_2,LOW);
          delay(200);
  }
  return distanciaXcYc;
}
void adelante(int vel_motor1, int vel_motor2){
  analogWrite(dirM1_1,vel_motor1);
  digitalWrite(dirM1_2,vel_apagado);
  analogWrite(dirM2_1,vel_motor2);
  digitalWrite(dirM2_2,vel_apagado);
}
void buscar(){
  analogWrite(dirM1_1, 110);
```

```
  digitalWrite(dirM1_2,LOW);
  analogWrite(dirM2_1, LOW);
  digitalWrite(dirM2_2,LOW);
}
void loop() {
  float actual = leerSensores();
  error_anterior=error;
  error = actual - referencia;
  proporcional = (float)Kp * error;
  integral += Ki*error;
  derivativo = Kd*error - error_anterior;
  float cpid = proporcional + integral + derivativo;
  vel_motor1 = vel_base - cpid;
  vel_motor2 = vel_base + cpid;
  if(vel_motor1 > vel_base ) vel_motor1 = vel_base - 10;
  if(vel_motor1 < 0 ) vel_motor1=0;
  if(vel_motor2 > vel_base ) vel_motor2 = vel_base + 30;
  if(vel_motor2 < 0 ) vel_motor2=0;
  if(distanciaXcYc != 0)
  {
          adelante(vel_motor1,vel_motor2);
          Serial.print("velM1 = " );
          Serial.print(vel_motor1);
          Serial.print("\t velM2 = ");
          Serial.print(vel_motor2);
          Serial.print("\t error = ");
          Serial.println(error);

        int sensor = analogRead(A0);
        Serial.println("Sensor = " );
        Serial.print(sensor);
        if( sensor >= 500)
        {
```

```
            analogWrite(dirM1_1, LOW);
                digitalWrite(dirM1_2,LOW);
                analogWrite(dirM2_1, LOW);
                digitalWrite(dirM2_2,LOW);
            delay(1000);
            servo1.write(160);
            delay(1000);
            analogWrite(A1, 255);
           analogWrite(dirM1_1, LOW);
               digitalWrite(dirM1_2, LOW);
               analogWrite(dirM2_1, LOW);
               digitalWrite(dirM2_2, LOW);
           delay(5000);
            }
    }
    else
    {
        analogWrite(A1, 0);
        servo1.write(110);
        buscar( );
    }
}
```

Apéndice C

Código G para el corte de las bases del robot

El corte de las bases que conforman el robot implementado se realizó a través de una máquina CNC, el código en el lenguaje G para obtenerlas es el siguiente.

```
G90 (programaci n absolutas)
G21 (trabajar con mil metros)
G00 Z20 (levantamos 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)
(RECTANGULOS)
G00 X100 Y100
```

```
G01 Z-8 F250.0 (bajamos herramienta con velocidad controlada, lenta)
G01 X100 Y85
G01 X85 Y85
G01 X85 Y20
G01 X15
G01 Y85
G01 X0
G01 X0 Y100
(ARCO)
G00 X0 Y100
G02 X100 Y100 I50 J0 F250.0
G00 Z20 (levantar herramienta)
M30
```

Referencias

[Almeida98] Almeida, L., Mota, A., y Fonseca, P. Comparing control strategies for autonomous line-tracking robots. *En Advanced Motion Control, 1998. AMC'98-Coimbra., 1998 5th International Workshop on*, págs. 542–547. IEEE, 1998.

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

[Cristiany12] Cristiany, T. y Antonio, J. *Construcción y localización en tiempo real de un robot móvil vía edometría en el seguimiento de trayectorias mediante control automático.* Tesis Doctoral, 2012.

[DelaEscalera15] De la Escalera, A. y Armingol, J. M. Visión por computador. *Fundamentos y métodos*, 2015.

[Haj-Ali04] Haj-Ali, A. y Ying, H. Structural analysis of fuzzy controllers with nonlinear input fuzzy sets in relation to nonlinear pid control with variable gains. *Automatica*, 40(9):1551–1559, 2004.

[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.

[Hirose07] Hirose, M. y Ogawa, K. Honda humanoid robots development. *Philo-*

sophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 365(1850):11–19, 2007.

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

[Instruments03] Instruments, T. Îa7800 series positive-voltage regulators. `https://www.sparkfun.com/datasheets/Components/LM7805.pdf`, 2003.

[Kuphaldt15] Kuphaldt, T. R. Pulse width modulation, 2015. [Internet, consultado el 07-octubre-2015].
URL `http://www.allaboutcircuits.com/textbook/semiconductors/chpt-11/pulse-width-modulation/`

[Kú15] Kú, J. M. C. Sensores efcto hall, inductivo, infrarrojo y magnetico, 2015. [Internet, consultado el 04-octubre-2015].
URL `http://www.alipso.com/monografias4/SENSORES_INDUCTIVO,_INFRAROJO,_MAGNETICO,_EFECTO_HALL/`

[Lázaro08] Lázaro, I., Castillo. Ingeniería de sistemas de control continuo. *UMSNH, COECyT Michoacán, FIE, Mexico*, 2008.

[LinuxCNC15] LinuxCNC. Emc2 g-code. http://linuxcnc.org/docs/2.1/html/gcode.html, 2015. Consulta: 1 de junio de 2015.

[Malpartida03] Malpartida, E. Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot. *Pontificia Universidad católica del Perú, Perú*, 2003.

[Manager14] Manager, C. Tecnologías actuales que parecen de ciencia ficción, 2014. [Internet, consultado el 29-septiembre-2015].
URL `http://sylarsystems.com/blog/?p=187`

[Mancha15] Mancha, U. C.-L. El servomotor, 2015. [Internet, consultado el 05-octubre-2015].

URL `http://www.info-ab.uclm.es/labelec/solar/electronica/elementos/servomotor.htm`

[Marinero02] Marinero, J. F. y Perán, J. Aplicaciones de la robótica: Últimas tendencias y nuevas perspectivas. *Dyna*, pág. 61, 2002.

[Moreno01] Moreno, M. A. Apuntes de control pid. *La Paz enero*, 2001.

[Museum15] Museum, C. H. Shakey, 2015. [Internet, consultado el 29-septiembre-2015].
URL `http://www.computerhistory.org/revolution/artificial-intelligence-robotics/13/289`

[Pardo12] Pardo, H. Q., Rolle, J. L. C., y Romero, O. F. Aplicación de un robot comercial de bajo coste en tareas de seguimiento de objetos application of a low cost commercial robot in tasks of tracking of objects. *Dyna*, 175:25, 2012.

[Santonja00] Santonja, R. y Sabater, J. Robots trepadores de estructura paralela dyna. 2000.

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

[Siegwart11] Siegwart, R., Nourbakhsh, I. R., y Scaramuzza, D. *Introduction to autonomous mobile robots*. MIT press, 2011.

[Sucar11] Sucar, L. E. y Gómez, G. Visión computacional. *Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, México*, 2011.

[Sun10] Sun, L. y Gan, J. Researching of two-wheeled self-balancing robot base on lqr combined with pid. *En Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, págs. 1–5. IEEE, 2010.

[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

[Umbaugh97] Umbaugh, S. E. *Computer Vision and Image Processing: A Practical Approach Using Cviptools with Cdrom*. Prentice Hall PTR, 1997.

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

[Wikipedia14] Wikipedia. Visión artificial — wikipedia: La enciclopedia libre, 2014. [Internet; descargado 26-febrero-2015].
URL {http://es.wikipedia.org/w/index.php?title=Visi\%C3\%B3n_artificial&oldid=79134046}

[Wikipedia15] Wikipedia. Puente h (electrónica). https://es.wikipedia.org/wiki/Puente_H_%28electr%C3%B3nica%29, 2015. Consulta: 28 De Mayo de 2015.