ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL PROVISTO DE VISIÓN ARTIFICIAL PARA RECONOCIMIENTO DE OBJETOS Y POSICIONAMIENTO DEL MÓVIL A LOS OBJETOS

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y CONTROL

PABLO ANDRÉS GARZÓN ORTEGA

pablo_garzon24@hotmail.com

JONATHAN GUILLERMO SOLA MARCILLO

jonathansola@hotmail.com

DIRECTOR: ANDRÉS FERNANDO CELA ROSERO, MSc

andres.cela@epn.edu.ec

Quito, Marzo 2016

DECLARACIÓN

Nosotros, Garzón Ortega Pablo Andrés y Sola Marcillo Jonathan Guillermo, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____ _____

Garzón Ortega Pablo Andrés Sola Marcillo Jonathan Guillermo

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Garzón Ortega Pablo Andrés y Sola Marcillo Jonathan Guillermo, bajo mi supervisión.

_____

Ing. Andrés Cela, MSc.

DIRECTOR DE PROYECTO

AGRADECIMIENTOS

A mis padres Fabián y Gladys, quienes con sus enseñanzas y esfuerzo constante me han dado las armas para poder enfrentar la vida, y quienes han sido y serán siempre mi respaldo. Los amo con todo mi corazón.

Al Ing. Andrés Cela, quien ha sabido ser más que un tutor un amigo, y ha inculcado en mí la pasión por la visión por computador.

A Jonathan Sola, mi compañero y amigo incondicional, quien con su paciencia, perseverancia y apoyo, ha sido un respaldo más para poder terminar con éxitos este objetivo de vida.

A la Ing. Pamela Zurita por haberme brindado tiempo, oportunidades, tolerancia y apoyo incondicional. Quien con su liderazgo se ha convertido en amiga, consejera y ejemplo de vida profesional y moral.

A mis compañeros de aula por todos los momentos gratos dentro de la universidad y fuera de ella. Por la amistad que hemos logrado a través de los años, les deseo éxitos en su vida.

A todas las personas que conforman mi ambiente laboral, por todos sus consejos, su paciencia y enseñanzas. Quienes han compartido su experiencia sin restricciones con el único afán de mostrarme el camino correcto de un profesional.

Pablo Garzón

AGRADECIMIENTOS

A Dios por haberme dado paciencia, fortaleza y perseverancia para lograr mis metas y retos a lo largo de mi vida, siendo mi guía y compañía obrando con honestidad y dedicación.

A mis padres Mariana y Orlando quienes me han dado los valores necesarios para ser una persona de bien y dedicada; quienes con su gran sacrificio me brindaron la oportunidad de seguir estudiando y ser alguien en la vida.

A mis hermanos Alexis y Carlos quienes con su conocimiento y paciencia supieron brindarme su apoyo en los momentos complicados en mi vida académica y personal. Quienes son mi ejemplo a seguir por su lucha, responsabilidad y perseverancia para conseguir los objetivos propuestos.

A Karina y Mercedes quienes han demostrado ser grandes hermanas para mí, de quienes he recibido el apoyo y aliento antes las adversidades que se presentaron en el transcurso de la realización de este proyecto.

A Msc. Andrés Cela al brindarme sus conocimientos, experiencias y amistad, al ser un apoyo incondicional en cada etapa de este proyecto.

A Patricia quien es mi complemento y apoyo incondicional para alcanzar mis objetivos, quien me da la fuerza para seguir adelante, quien forma parte de mi vida y está siempre a mi lado.

A mis amigos que formaron gran parte de mi vida académica al brindarme su apoyo, compañerismo y conocimientos hasta el momento de lograr mi meta, en especial a Pablo quien es parte fundamental para el cumplimiento de este sueño que ahora compartimos juntos.

Jonathan Sola

DEDICATORIA

A nuestras familias que ven en la culminación de esta meta el reflejo de todas sus enseñanzas, valores y esfuerzo. Quienes seguirán siendo el pilar fundamental de nuestras vidas.

Pablo y Jonathan

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Este trabajo presenta el diseño y construcción de un robot móvil provisto de una cámara, usada como sensor para reconocer objetos por color y forma, para determinar por medio de un algoritmo la posición de cada objeto con respecto al robot y transmitir esta información al dispositivo como puede ser una placa arduino.

Los algoritmos de medición de distancias están basados en la geometría de semejanza de triángulos, tomando como referencia las características del modelo de cada objeto.

Luego de aplicar diferentes algoritmos de procesamiento de imágenes reconocimiento, segmentación, filtros, etc. Se obtiene el área de los objetos y posteriormente la distancia de los objetos.

La estructura del robot, el cual tiene el modelo uniciclo con tracción diferencial fue diseñada en el programa inventor, la movilidad es controlada por medio de dos motores.

Las señales de control (PWM) para los motores provienen del controlador PID, el cual fue ajustado por el método de Ziegler y Nichols. La referencia del robot es el centro de la imagen. Se calcula el ángulo formado entre el centroide del objeto detectado y el centro de la imagen, siendo este ángulo el error que ingresará al controlador.

El robot está diseñado de tal forma que tenga una autonomía aproximadamente de 4 horas, para ello se ha usado sistemas independientes de baterías.

Los parámetros de seguimiento pueden ser enviados y monitoreados a través de la interfaz gráfica de raspberry Pi, esto se realiza por medio de una red de conexión inalámbrica entre la tarjeta SBC y la PC

Finalmente, en este proyecto se ha logrado implementar un sistema de visión dentro de un sistema embebido, que no cuenta con las características necesarias de procesamiento y memoria que normalmente se usan en un proyecto de visión por computador.

PRESENTACIÓN

En el presente proyecto se implementa un robot dotado de visión artificial cuyo objetivo es la identificación y seguimiento de un objeto por color y forma mediante la utilización de una plataforma comercial de 32bits dotada de software libre. Este sistema mide distancias de profundidad, altura y ángulo de desviación del objeto detectado, con referencia al robot, y transmite esta información usando el protocolo I2C.

En el primer capítulo se puede destacar los conceptos teóricos fundamentales que ayudan a llevar acabo la realización de este proyecto como: la visión por computadora, el control y el hardware utilizado.

En el segundo capítulo se presenta el diseño y la construcción del robot, considerando los espacios físicos de los dispositivos que conforman el mismo. Además se describen de forma general las características principales de cada uno de ellos, los cuales ayudan al correcto funcionamiento del robot.

En el tercer capítulo se presentan los algoritmos empleados para la captura, segmentación, identificación, reconocimiento de objetos basados en las librerías de OpenCv, medición de distancias tanto de profundidad, altura y ángulo de desviación. Además se presenta el diseño del controlador de los motores el cual depende del ángulo de desviación del objeto detectado con referencia al robot para el seguimiento del mismo.

En el cuarto capítulo se presentan las pruebas realizadas y los resultados obtenidos. Además se analiza los errores y se describe las soluciones adoptadas.

En el quinto capítulo se muestra las conclusiones y recomendaciones basadas en los resultados obtenidos en este proyecto.

CAPÍTULO 1

MARCO TEÓRICO

En este capítulo presenta la información necesaria para entender todos los cálculos, algoritmos y demás procesos realizados para la realización de este proyecto, empezando por las definiciones básicas de visión por computador, descripción del software, la tarjeta SBC y controlador utilizado

1.1 VISIÓN POR COMPUTADOR

La visión por computador también conocida como visión artificial busca interpretar características y propiedades de una imagen 2D o 3D, la cual puede ser cambiante, puede haber sido captada por una o varias cámaras o proceder de sensores de características no visuales como acústicos, térmicos, táctiles, etc. [1]

1.1.1 DEFINICIÓN

Basado en las ideas presentadas previamente, la visión por computador es: “El análisis de una imagen para recuperar información espacial y temporal de la misma”. [2]

1.1.2 APLICACIONES

- Reconocimiento de caracteres (OCR).
- Inspección Visual.
- Ventas (verificación de productos).
- Reconstrucción de modelos 3D.
- Imágenes Médicas.
- Seguridad en automóviles (peatones, carril, cansancio, etc).
- Acople de movimiento (acopla gráficos por computadora a sistemas de video real).
- Captura de movimiento (capturar movimientos grabados para transferirlos a un modelo físico del objeto grabado).
- Vigilancia.
- Biométrica (caras, huellas dactilares, retina). [2]

1.1.3 DEFINICIONES PRELIMINARES

A continuación se presentan algunos conceptos que ayudarán al entendimiento del proyecto realizado.

1.1.3.1 Luz Visible

La luz está constituida por ondas luminosas de diferentes longitudes de onda. Las longitudes de onda visibles por el ojo humano constituyen el denominado espectro visible, el cual va desde los 400nm hasta los 700nm.

El ojo humano al percibir estas longitudes de onda envía estímulos al cerebro, el cual produce sensaciones como el brillo y el color de los objetos. [3]

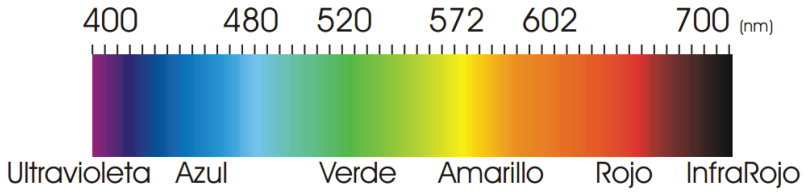


Figura 1.1. Espectro visible de la radiación electromagnética [3]

1.1.3.2 Flujo Radiante

Es la cantidad de energía emitida por ondas electromagnéticas provenientes de una fuente. Su unidad de medida es el vatio. [3]

1.1.3.3 Flujo Luminoso

Es la cantidad de flujo radiante recibido por el ojo humano tras las perdidas por conducción y la cantidad de flujo luminoso que se encuentra en el espectro no visible. Su unidad es el lumen. [3]

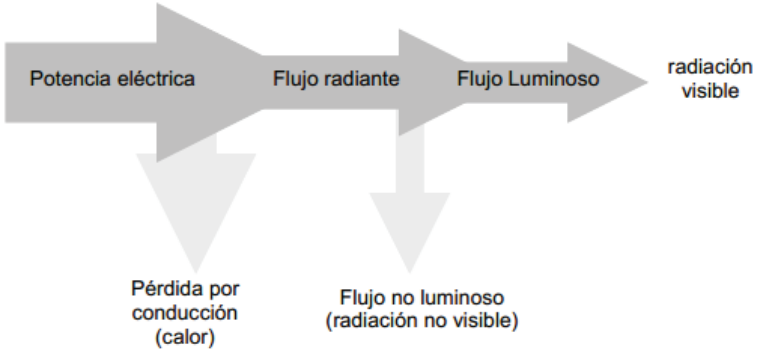


Figura 1.2. Diagrama de pérdidas de flujo luminoso [3]

1.1.3.4 Intensidad Luminosa

Es la cantidad de flujo luminoso emitido por unidad de ángulo sólido. Su unidad es la candela.

La intensidad luminosa no depende de la distancia a la fuente de luz, varía según la orientación de la medición. [3]

1.1.3.5 Bastones

Son células especializadas en la retina, que se encuentran en un número mayor a 100 millones. Detectan y miden el brillo de haces luminosos.

1.1.3.6 Conos

Son células especializadas en la retina con un número de 6 millones, son capaces de variar su comportamiento frente a los cambios de longitud de onda de un haz luminoso (color). Tienen una sensibilidad menor a los bastones. [3]

Existen tres tipos de conos:

- Tipo S (Short): sensibles a longitudes de onda corta (azules).
- Tipo M (Medium): sensibles a longitudes de onda media (verdes).
- Tipo L (Large): sensibles a longitudes de onda larga (rojo).

1.1.3.7 Brillo

Es la cantidad de flujo luminoso recibido por unidad de superficie [4]

1.1.3.8 Tono o Matiz

Es la relación entre las activaciones de distintos tipos de conos al percibir una imagen, depende de la longitud de onda dominante, es decir, aquella que presente más energía en el diagrama espectral. [4] , [3]

1.1.3.9 Contraste

Es la diferencia entre la luminosidad y tono entre zonas oscuras y claras de una imagen. [4]

1.1.3.10 Saturación

Es la proporción entre la longitud de onda dominante y las demás longitudes de onda de una imagen. Se la observa como la viveza o palidez del color [4], [3]

1.1.3.11 Pixel

Proviene del nombre en inglés “picture element”. Es un par ordenado [2]

$$e = (p, c) \tag{1.1}$$

Dónde:

p: Vector de posición, de una rejilla de dos dimensiones denominada G, $G^2 = [0 \dots a_1 - 1] \times [0 \dots a_2 - 1]$

c: Vector d-dimensional $c \in \mathcal{R}^d$. El cual contiene las características del pixel como: valor de energía, color, textura, valor de gris, etc. [2]

1.1.3.12 Imagen

Puede vérsela como una señal, conjunto, función, etc. De la siguiente manera:

1.1.3.12.1 Conjunto

La imagen I es un conjunto finito, no vacío de pixeles que cumplen las siguientes propiedades [2]:

- Dos pixeles con posiciones diferentes, son iguales si y solo si sus vectores de características son iguales. [2]

$$Para: e_i = [p_i, c_i] \in I \; y \; e_j = [p_j, c_j] \in I, \quad p_{\boldsymbol{i}} = p_{\boldsymbol{j}} \quad \Rightarrow \quad c_i = c_j \tag{1.2}$$

- La posición de un pixel siempre estará dentro de la rejilla de la imagen. [2]

$$p_{\boldsymbol{i}} \in G^2 \tag{1.3}$$

- El número de elementos en la imagen denominado cardinalidad, será igual a la cardinalidad de la rejilla G^2. [2]

$$|I| = |G^2| \tag{1.4}$$

1.1.3.12.2 Función

Es una función vectorial que mapea la rejilla G^2 a un vector del espacio d-dimensional de características [2]

$$I \; es \; f_I: G^2 \rightarrow \mathcal{R}^d, de \; modo \; que, e_i \in I \Leftrightarrow e_i = [p_i, f_I(p_i)] \tag{1.5}$$

Es una función vectorial, que asume para cada posición un vector de valores denominado canal de la imagen. [2]

$$f_I(p) = [f_1(p), \dots, f_d(p)]^T \tag{1.6}$$

Dónde:

$f_i(p)$: Es cada uno de los canales de la imagen.

1.1.3.13 Región

Es un subconjunto no vacío de la imagen.

$$I = \mathcal{R} \subseteq I, R \neq \emptyset \tag{1.7}$$

Matemáticamente no necesita ser conexa, es decir dos partes visiblemente independientes de una imagen pueden ser asignadas a una misma región. Sin embargo se puede agregar información posterior que restrinja la región a partes conexas. [2]

1.1.3.14 Fondo

El fondo de una imagen se compone de todo aquello que no forma parte del objeto a reconocerse dentro de la imagen.

1.1.3.15 Objeto

Entidad del mundo real que tiene un significado dentro del contexto de una aplicación. [2]

1.1.3.16 Histograma

El histograma es una herramienta cualitativa y cuantitativa de una imagen. Es un gráfico de la distribución de las intensidades de gris en una imagen (eje x), y la cantidad de pixeles que se encuentran dentro de cada nivel de gris (eje y). Este gráfico es bidimensional. [5]

Este gráfico puede proveer información de brillo, contraste, así como el rango dinámico de una imagen. [5]

Figura 1.3. Histograma de una imagen

1.1.4 MODELO FISIOLÓGICO DE VISIÓN

1.1.4.1 El sistema visual

Está conformado por el ojo, que es un órgano que transforma la luz capturada en un impulso neuronal para su procesamiento en el cerebro, el cual genera sensaciones, completando así el proceso de percepción visual. [3]

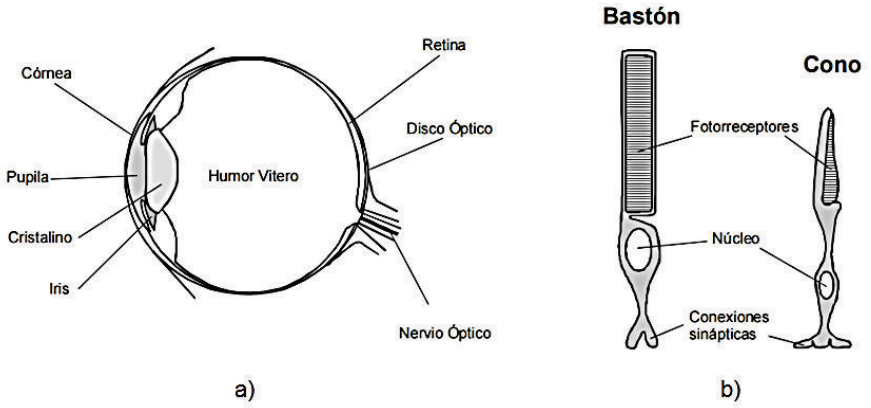


Figura 1.4 a) Sección del ojo humano. b) Visión esquemática de las células fotoreceptoras. [3]

1.1.4.2 Percepción acromática

La sensación que producen los bastones se relaciona con los siguientes fenómenos: [3]

- Sensibilidad a la intensidad:

  Permite distinguir un nivel de intensidad de otro. Si el número de intensidades percibidas a la vez por el ojo es mayor a 24 tonos se pierde la mayoría de esta sensibilidad. [3]

  La intensidad percibida no es lineal. Esto puede observarse en la Figura 1.5. La curva A representa el brillo apreciado el brillo apreciado en función con el brillo físico reflejado por un pigmento. Se aprecia que el humano es capaz de distinguir pigmentos de intensidades poco diferentes (como a_1 y a_2). Pero cuando existe contraste acentuado la apreciación cambia.

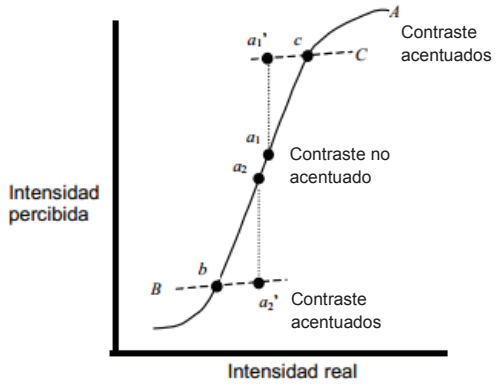


Figura 1.5. La línea A representa la relación entre el brillo distinguido por el ojo humano y el nivel de brillo real. [3]

La siguiente imagen demuestra que cuando hay muchos niveles involucrados se disminuye la precisión en la detección del brillo. Se necesitan periodos de adaptación para cada situación. [3]

Figura 1.6. El color gris del cuadrado interior de la figura de la derecha parece más oscuro que el cuadrado interior de figura de la izquierda, a pesar de que ambos están tintados con el mismo gris. [3]

- Inhibición lateral:

  Debido a las células de la retina, que al detectar un nivel de intensidad inhiben a las células cercanas, produciendo perturbaciones en cada cambio de intensidad como se observa en la Figura 1.7. [3]

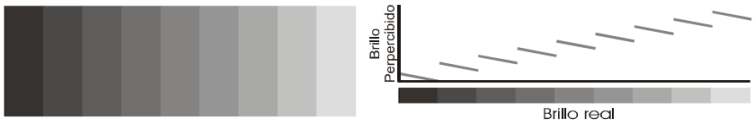


Figura 1.7. La tonalidad de cada una de las franjas verticales de la figura de la izquierda es uniforme. El brillo percibido para cada banda se refleja en el diagrama de la derecha [3]

1.1.4.3 Percepción cromática

Se debe a la sensación percibida por los conos, la cual da al cerebro las señales que se interpretaran como colores de acuerdo a las diferentes longitudes de onda de los colores azules, verdes y rojos.

1.1.5 MODELO DE VISIÓN

La visión por computador intenta reproducir el comportamiento que tiene el cerebro al procesar una imagen captada por la vista, para esto define cuatro fases:

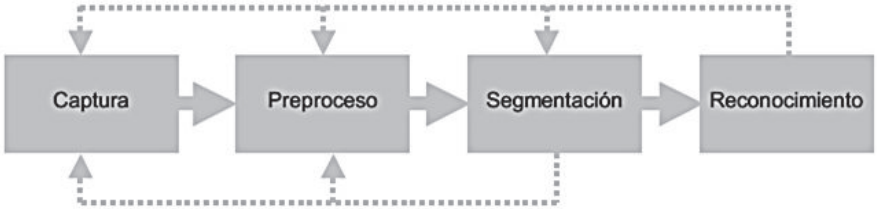


Figura 1.8. Etapas típicas en un sistema de visión por computador [3]

- **Captura:** esta fase es sensorial, se adquieren las imágenes mediante un sensor y se digitalizan.
- **Preproceso:** se realiza el filtrado de la imagen para poder dar realce a elementos importantes de la misma o eliminar elementos que no se desee analizar.
- **Segmentación:** se apartan los elementos que interesan en la imagen para poder comprenderla.
- **Reconocimiento:** se distinguen los elementos que han sido apartados de acuerdo a parámetros y características previamente establecidas.

Este modelo no siempre es secuencial, en ocasiones se hace necesario regresar a etapas anteriores con el fin de corregir hipótesis falsas que ayude en el desarrollo de las mismas.

La fase de pre-proceso está constituida a su vez por el modelo de visión general de Marr-Palmer. Palmer basado en el modelo de David Marr presenta un modelo de visión que permita analizar los problemas a nivel computacional.

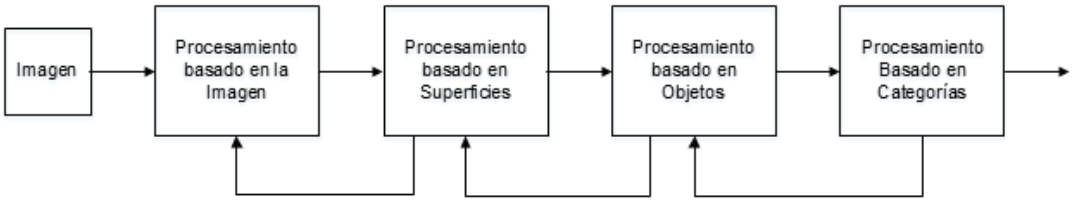


Figura 1.9. Modelo general de visión de Marr-Palmer

Con este modelo podemos observar los problemas de visión a diferentes niveles.

- Procesamiento en el nivel de imagen: se tiene el contenido de cada pixel (posición, valor de color, etc.)
- Procesamiento basado en superficies: se introduce información adicional para regularizar el problema, como configuraciones de luz, colores, etc.
- Procesamiento basado en objetos: igual al anterior se introducen información o conocimientos para regularizar los problemas, como que superficies con ciertas características forman una imagen.
- Procesamiento basado en categorías: busca relaciones entre objetos en una imagen.

1.1.6 SEGMENTACIÓN

Es el proceso de tomar una imagen y separarla en regiones que tienen un interés particular para una aplicación.

Debido a esto se presenta la paradoja de la visión por computador:

"Para segmentar objetos es necesario reconocerlos, y para reconocerlos, es necesario segmentarlos". [2]

1.1.6.1 Segmentación supervisada:

Un ser humano interactúa con el proceso para determinar que partes del objeto son de interés. [2]

1.1.6.2 Segmentación no supervisada

No requieren de ninguna información adicional dada por un ser humano. [2]

1.1.6.3 Segmentación a nivel de objeto

La segmentación S de una imagen I es un conjunto de regiones de la imagen que satisface:

1.

$$|L(\mathcal{R}_i)| = 1 \; para \; \mathcal{R}_i \; \in S, i = 1 \ldots n_{opt} \tag{1.8}$$

Esta condición establece que el operador de reconocimiento (L) para una región es un solo objeto.

Se asegura que no hay varios objetos para una región.

n_{opt} es el valor óptimo de regiones etiquetadas con el operador de reconocimiento para un objeto a segmentar.

2. $n_{opt} = |S| \; es \; mínimo$ (1.9)

Esta condición indica que no se está sobre-segmentando. [2]

1.1.6.4 Segmentación a nivel de imagen

Se basa en un criterio de homogeneidad o uniformidad. Se definen regiones como un grupo de pixeles que comparten una cierta característica. Suponiendo que los objetos se componen de superficies homogéneas.

Se define la segmentación basada en imagen en términos de homogeneidad H_l y adyacencia $\mathcal{A}$.

La segmentación a nivel de imagen satisface que para cada región $\mathcal{R}_i \in S_i$ $H_l(\mathcal{R}_i) = verdadero$, y $H_l(\mathcal{R}_i \cup \mathcal{R}_j) = falso$ para $\mathcal{A}(\mathcal{R}_i, \mathcal{R}_j) = verdadero$, es decir, para cada región en un grupo de regiones el criterio de homogeneidad da verdadero si se analiza cada región por separado, pero si se unen dos regiones adyacentes este criterio da falso. [2]

Con esto se logra prevenir la sobre-segmentación.

Cada método de segmentación provee su manera particular de definir la homogeneidad e impone sus restricciones para una región.

1.1.6.5 Métodos de segmentación a nivel de imagen

La segmentación con base al espacio de características toma en cuenta la definición de pixel $e_i = (p_i, c_i)$, aunque se ignora la información p (posición del pixel), se muestran a continuación los siguientes métodos:

1.1.6.5.1 Umbralización

- Se parte la imagen en dos regiones: el objeto y el fondo
- Todos los pixeles que tengan un valor de características (c) que sea menor a un umbral (T) se asignan al fondo y el resto al objeto.

- El umbral o threshold puede ser provisto por el usuario o estimado automáticamente (algoritmo de Otsu). [2]

1.1.6.5.2 Histogramas

En imágenes en grises, con un solo histograma, se hace la detección de máximos y así se determinan diferentes clases, simplemente separando los máximos.

Figura 1.10. Imagen en escala de grises con su respectivo histograma. [2]

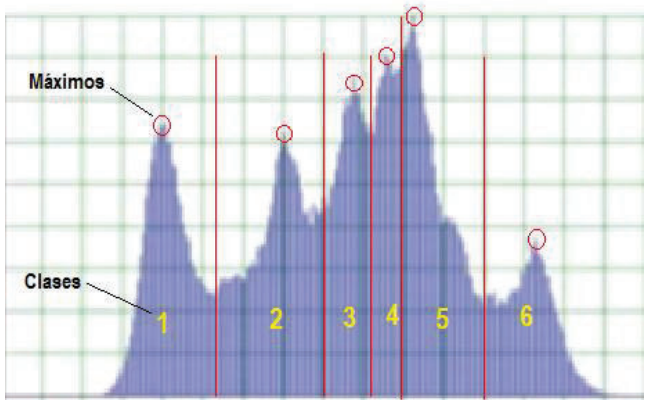


Figura 1.11. Segmentación de imagen por histograma, reconocimiento de máximos y clases. [2]

Este método de segmentación se lo realiza cuando se quiere segmentar objetos muy claros de una imagen muy oscura. [2]

1.1.7 ESPACIOS DE COLOR

1.1.7.1 Espacio RGB

Es un espacio de color formado por tres planos. Cada plano representa un color primario (Red, Gren, Blue) que puede tener valores dentro de un rango de [0,1].

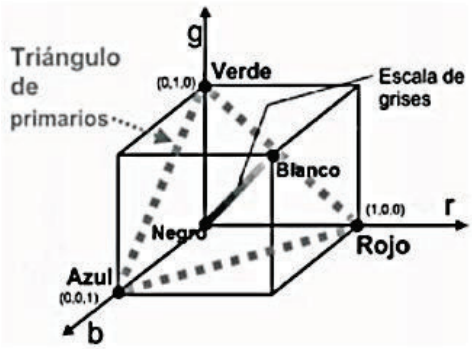


Figura 1.12. Representación del espacio de color RGB [6]

Cada plano puede ser representado a su vez como canales de un pixel asignando un 1 byte a cada uno, lo que equivale a 256 tonalidades de color. Donde 0 indica una mínima intensidad del color y el 255 indica una máxima intensidad del color.

Como son 3 bytes que se utilizan para representar a un pixel se tendría 256^3 = 16777216, lo que equivale al volumen del cubo. [6]

Por lo tanto en este espacio el pixel tendría la forma:

$$e = (p, R, G, B) \tag{1.10}$$

Dónde:

p: es el par ordenado de la posición del pixel en la imagen.

Las imágenes en este modelo se forman por la combinación en diferentes proporciones de cada uno de los colores primarios RGB. [7]

1.1.7.2 Espacio HSV

Este espacio de color fue creado para representar una imagen combinando tres valores: Tono (Hue), la saturación (Saturation) y el valor o brillo (Value). [8]

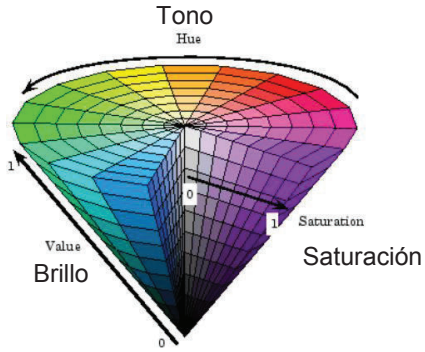


Figura 1.13. Representación gráfica del espacio HSV [9]

- Hue: Color puro. Está comprendido entre el rango de [0,360º]
- Saturation: degradación con blanco (claro u obscuro). Comprendido en entre el rango de [0-100%]
- Value: brillo o mate. Comprendido entre el rango de [0-100%] [9]

1.1.7.2.1 Conversión RGB-HSV

$$H = H_1, \qquad Si\ B \leq G$$
$$H = 360º - H_1 \qquad Si\ B > G \tag{1.11}$$

$$H_1 = \cos^{-1}\left(\frac{\frac{1}{2}\big((R-G)+(R-B)\big)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \tag{1.12}$$

$$S = \frac{M-m}{M} \quad y \quad V = \frac{M}{255} \tag{1.13}$$

$$M = \max(R,G,B)$$
$$m = \min(R,G,B) \tag{1.14}$$

1.2 SOFTWARE

1.2.1 PYTHON

Es un lenguaje de programación optimizado para aumentar la productividad del programador, legibilidad de código y calidad de software.

Tiene estructuras de datos de alto nivel y un efectivo enfoque en la programación orientada a objetos. La elegante sintaxis y dinámica de escritura conjuntamente

con su interpretación natural lo convierten en un lenguaje ideal para el desarrollo de aplicaciones rápidas en diferentes plataformas de software libre.

Esta librería es de libre adquisición desde su sitio web https://www.python.org/, la cual contiene programas, herramientas e información adicional acerca de todos los módulos de Python

La interfaz de Python puede ser extendida con nuevas funciones y tipos de datos implementados en C o C++. [10]

Figura 1.14. Logotipo del programa Python

1.2.1.1 Python 2.x

Dentro de la versión 2.x la mejor publicación es la 2.7. Aunque a pesar de que ya se hizo el lanzamiento de la versión 3.x, la versión 2.x continua recibiendo mejoras constantemente. Esto significa que Python 2.7 permanecerá aún por mucho tiempo, entregando una plataforma con soporte estable para producción de sistemas que no tienen todavía soporte para Python 3.

Algunas de los efectos del soporte para Python 3 son las siguientes.

- Recibir actualizaciones de seguridad y reparación de conflictos hasta el 2020.
- Los paquetes que se desarrollen para Python 3 podrán actualizar también a Python 2.7, lo cual facilitará con el tiempo la migración a la versión más reciente.
- El enfoque de desarrollo de los paquetes no siempre funcionará correctamente en la versión 2.7, como por ejemplo los relacionados con seguridad de redes. En casos excepcionales no se podrá manejar las actualizaciones de paquetes en PyPi. [10]

1.2.1.2 Python 3.x

La cobertura ha sido aumentada a por un nuevo rendimiento y crecimiento de sintaxis. [11]

Una lista parcial de las mejoras que fueron desarrolladas son:

- La sintaxis para un conjunto literal es mutable.
- Diccionario y conjunto de comprensiones.
- Múltiple manejo de contexto con una simple frase.
- Una nueva versión de la librería IO, reescrita en C para desarrollo.
- Un pequeño subconjunto del módulo importlib.
- Conversiones correctamente redondeadas de punto flotante a string y viceversa.
- Otras. [10]

1.2.2 OPENCV

Open CV es una librería de visión por computador de fuente abierta, escrita en C y C++. Tiene soporte para varios sistemas operativos como: Windows, Linux, Mac, Android y varias interfaces como: Python, Matlab y otros lenguajes.

Esta librería se puede adquirir de forma gratuita desde su sitio web: www.opencv.org

Fue diseñado para eficiencia computacional con un fuerte enfoque en aplicaciones de tiempo real.

Su objetivo es proporcionar infraestructura sencilla de usar, de manera que el usuario pueda realizar aplicaciones de visión por computador bastante sofisticados.

Contiene además una librería de propósito general, Machine Learning Library (MLL) que es en su mayoría usada para tareas de visión que están dentro del núcleo de OPENCV, sin embargo es suficiente usarla para cualquier problema de aprendizaje automático. [12]

Figura 1.15. Logotipo de la librería opencv

1.2.2.1 Ventajas del uso de OpenCV

Existen importantes objetivos para OpenCV que lo convierten en la primera opción para desarrollo de aplicaciones de visión por computador:

- Avanzar en la investigación de visión por computador proporcionando código abierto.
- Difundir el conocimiento de visión, proporcionando una estructura común para cualquier interfaz de programación, por lo que el código sería más fácil de leer y transferir.
- Optimizar la disponibilidad del código en forma gratuita, de manera que las aplicaciones no requieran de una licencia comercial. [12]

1.3 RASPBERRY PI

En 2006 el ingeniero Eben Upton fabricante de chips Broadcom se impuso el reto de diseñar microordenadores, para que tanto niños como adultos puedan realizar proyectos computacionales de programación. Reunió un gran grupo de ingenieros, programadores de Broadcom, la Universidad de Cambridge y otras corporaciones y organizaciones educacionales para así crear la fundación Raspberry Pi. [13]

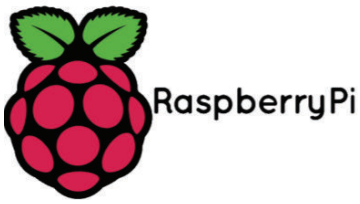


Figura 1.16. Logotipo Fundación Raspberry Pi

La fundación raspberry pi ha creado tarjeta computacionales para el aprendizaje de sistemas de control a bajo costo y alta gama de características, que pueden competir con herramientas de control como Arduino, Pics, Atmega con la particularidad del uso de un sistema operativo con software libre.

Raspberry Pi es un computador personal y no solamente un microcontrolador. Este computador personal mejora el siguiente procesamiento de datos.

- Input: el computador recibe instrucciones y datos de un usuario o aplicación.

- Proceso: el computador desarrolla acciones pre-programadas sobre las entradas.
- Output: El computador muestra los resultados del proceso en una o varias maneras, según la aplicación del usuario. [13]

1.3.1 MODELOS DE RASPBERRY PI

1.3.1.1 Modelo A

Es el modelo menos respetado de Raspberry Pi, posee 256MB de RAM, un puerto USB, pero no posee puerto Ethernet. Pensado para proyectos embebidos debido a su menor peso y consumo de poder. Es usado comúnmente en robótica para proyectos que no requieran de mucho procesamiento o con un adaptador wifi es perfecto para un centro multimedia. [14]

Figura 1.17. Raspberry Pi Modelo A

1.3.1.2 Modelo A+

Es el modelo de más bajo costo. Reemplazo al modelo A en noviembre del 2014, entre sus características más importantes se observan las siguientes:

- Aumento de pines GPIO a 40
- Socket para tarjeta SD cambia a la versión micro SD
- Reducción del consumo de poder entre 0.5W y 1W.
- Mejora de audio debido a una fuente dedicada de bajo ruido.
- Aproximadamente 2cm más pequeña que el modelo A. [14]

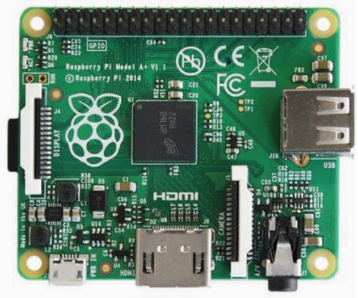


Figura 1.18. Raspberry Pi Modelo A+

1.3.1.3 Modelo B

Posee 512MB de RAM, dos puertos USB 2.0 y un puerto Ethernet. [14]

Figura 1.19. Raspberry Pi Modelo B

1.3.1.4 Modelo B+

Es la revisión final de la Raspberry Pi original. Reemplazó al modelo B en Julio del 2014, entre sus características más importantes se observan las siguientes:

- Aumento de pines GPIO a 40.
- Aumento a 4 puertos USB 2.0
- Socket para tarjeta SD cambia a la versión micro SD.
- Reducción del consumo de poder entre 0.5W y 1W-
- Mejora de audio debido a una fuente dedicada de bajo ruido.
- Distribución adecuada para agujeros de montaje. [14]

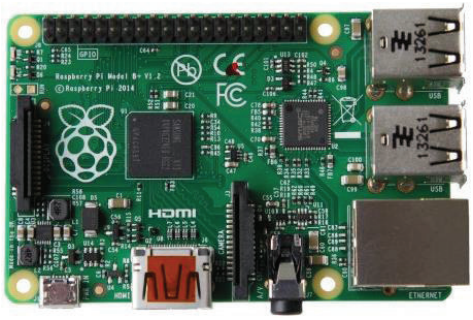


Figura 1.20. Raspberry Pi Modelo B+

1.3.1.5 Raspberry Pi 2 Modelo B

Es la segunda generación de Raspberry Pi, que reemplazó al modelo B+ en febrero del 2015. Entre sus características más relevantes están:

- Procesador de cuatro núcleos a 900 MHz ARM Cortex-A7.
- 1GB de RAM.
- Todas las características del modelo B+. [14]

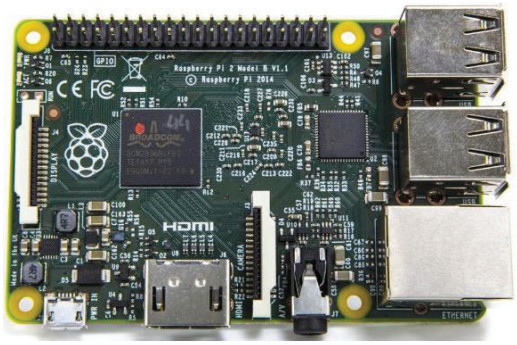
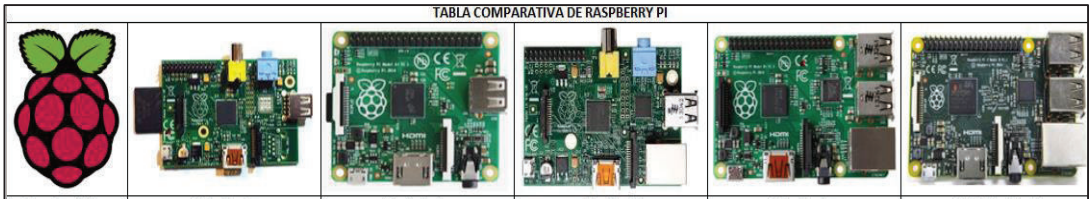


Figura 1.21. Raspberry Pi 2 Modelo B

1.3.1.6 Tabla comparativa de modelos

TABLA COMPARATIVA DE RASPBERRY PI



Características	Modelo A	Modelo A+	Modelo B	Modelo B+	Pi 2 Modelo B
SoC	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836
CPU	700 MHz ARM1176JZF-S	700 MHz ARM1176JZF-S	700 MHz ARM1176JZF-S	700 MHz ARM1176JZF-S	900 MHz Quad-core ARM Cortex-A7
GPU	Video Core IV	Video Core IV	Video Core IV	Video Core IV	Video Core IV
RAM	256 MB	256 MB	512 MB	512 MB	1 GB
USB	1	1	2	4	4
Video	RCA, HDMI	Jack, HDMI	RCA, HDMI	Jack, HDMI	Jack, HDMI
Audio	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI
Boot	SD	Micro SD	SD	Micro SD	Micro SD
Red			Ethernet 10/100	Ethernet 10/100	Ethernet 10/100
Consumo	300 mA / 1,5 W / 5V	400 mA / 2 W / 5V	700 mA / 3,5W / 5V	500 mA / 2,5 W / 5V	800 mA / 4 W / 5V
Alimentación	Micro USB / GPIO	Micro USB / GPIO	Micro USB / GPIO	Micro USB / GPIO	Micro USB / GPIO
Tamaño	85,6 X 53,98 mm	65 X 56 mm	85,6 X 53,98 mm	85 X 56 mm	85 X 56 mm
Precio	25 $	20 $	35 $	35 $	35 $

Figura 1.22. Tabla comparativa de los modelos de Raspberry Pi

1.3.2 SISTEMAS OPERATIVOS SOPORTADOS POR RASPBERRY PI

Los sistemas operativos que tiene soporte para Raspberry Pi son los siguientes:

- Raspbian
- Arch Linux ARM
- Fedora Remix
- Occidentalis
- Open ELEC
- RaspBMC
- RISC OS

1.3.2.1 Raspbian

Es el sistema operativo recomendado para Raspberry Pi. Se trata de una versión de Linux llamada Debian, y adaptada para trabajar en Raspberry Pi. De ahí la procedencia de su nombre.

Figura 1.23. Raspbian como resultado de la fusión de Raspberry PI y Debían

- Posee uno de los más poderosos y flexibles manejadores de paquetes.
- Presenta una interfaz gráfica amigable con el usuario. [13]

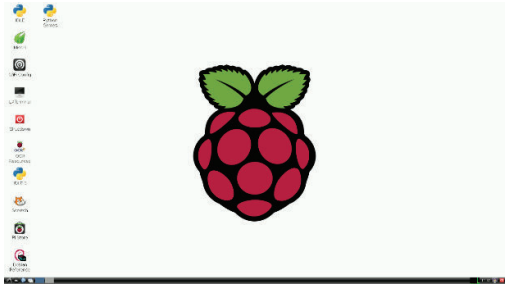


Figura 1.24. Pantalla de inicio del sistema operativo Raspbian

1.3.3 DESCRIPCIÓN DEL HARDWARE DE RASPBERRY PI

El circuito impreso de la Raspberry Pi está compuesto por pistas de cobre, resistencias, capacitores, transistores y circuitos integrados. Entre estos elementos el más importante es el procesador llamado Broadcom BCM2835, los elementos tienen una terminología para poder distinguirlos en el circuito impreso, como se observa en la Figura 1.25 [13]

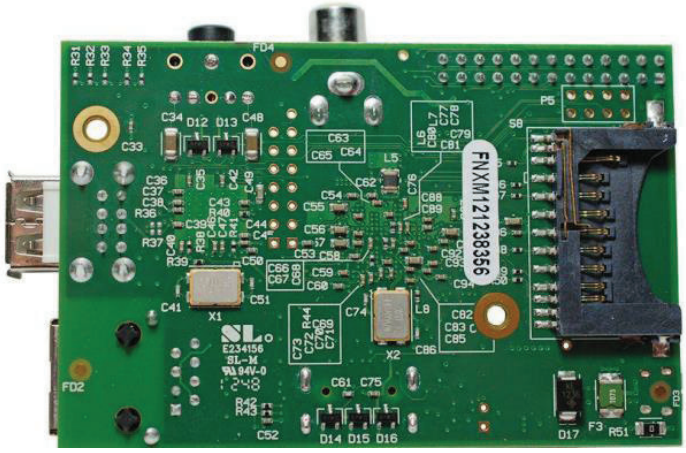


Figura 1.25. Lado posterior de la Raspberry Pi Modelo B [13]

1.3.3.1 Descripción de Periféricos

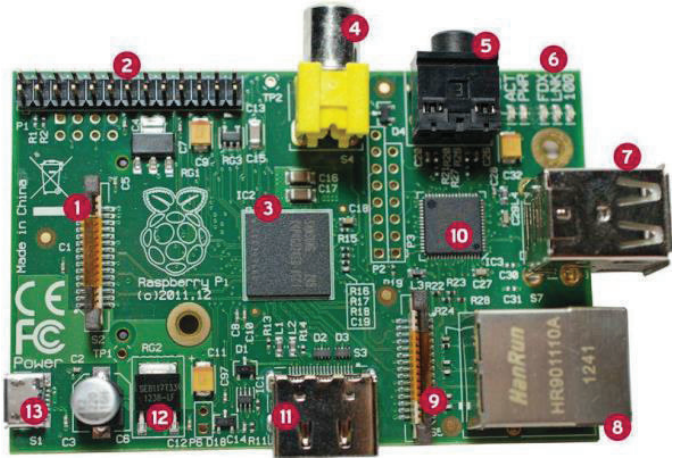


Figura 1.26. Lado frontal de la Raspberry Pi Modelo B [13]

1.3.3.1.1 (#1) Video DSI

DSI (Display Serial Inteface), es un estándar para pantallas LCD de bajo costo. Se encuentra conectado directamente a la unidad de procesamiento gráfico GPU. El DSI define el protocolo entre la pantalla y el procesador de video, y puede ser usado conjuntamente con un protocolo de comunicación I2C para darle la capacidad de una pantalla touch.

El tipo de pantalla está limitado por el GPU ya que no se cuenta con los driver necesarios para cualquier tipo de sistema. [15]

1.3.3.1.2 (#2) GPIO

Es un puerto de entradas y salidas de propósito general, dependiendo del modelo de Raspberry Pi este puerto puede tener entre 26 pines y 40 pines. Permite a la Raspberry Pi Interactuar con otros dispositivos electrónicos. [15]

Este puerto será estudiado más a fondo en la sección 1.3.4

1.3.3.1.3 (#3) CPU/GPU/RAM

Este procesador formado por cuatro núcleos, es la versión más reciente BCM2836 para la Raspberry Pi 2 Modelo B, tiene el doble de memoria RAM que su predecesor BCM2835. Lo especial de este procesador son sus paquetes de video y multimedia, incremento en gráficos que son una solución multimedia altamente competitiva.

Algunos de los aspectos más relevantes de este procesador se muestran a continuación:

- CPU: 900 MHz quad-core ARM Cortex A7 (ARMv7)
- SoC: Broadcom BCM2836 (CPU, GPU, DSP, SDRAM)
- GPU: Broadcom VideoCore @ 250MHz, OpenGL ES 2.0 (24 GFLOPS), 1080p30 MPEG-2 y decodificador VC-1.
- Memoria: 1GB (compartida con la GPU). [16]

1.3.3.1.4 (#4) RCA Video

Salida de video análoga que fue colocada en todos los modelos de la Raspberry Pi 1, aunque a partir del Modelo B y la Raspberry Pi 2 modelo B fue retirado. La fundación Raspberry Pi pensó en la salida RCA debido a su utilidad y a que con un amplio rango de dispositivos ha tenido un excelente desempeño.

El archivo config.txt contiene los parámetros para poder configurar la salida de video RCA, la muestra estas configuraciones [15]

Tabla 1.1. Modos de video soportados por la salida RCA

Configuración en config.txt	**Descripción de la configuración**
sdtv_mode=0standard	NTSC
sdtv_mode=1	NTSC-J
sdtv_mode=2	Standard PAL
sdtv_mode=3	PAL-M
sdtv_aspect=1	Aspect ratio of 4:3
sdtv_aspect=2	Aspect ratio of 14:9

1.3.3.1.5 (#6) LEDs de estado

Todos los modelos de la Raspberry Pi 1 poseen 5 leds que indican su estado de funcionamiento (ACT, PWR, FDX, LNK, 100), aunque a partir de la Raspberry Pi 2 Modelo B solo se tiene 2 Leds de estado (ACT, PWR). [13]

Tabla 1.2. Descripción de los Leds de Estado de la Raspberry [13]

Led	Color	Descripción
ACT	Verde	Acceso a la tarjeta SD
PWR	Rojo	Alimentación de 3,3 v (OK)
FDX	Verde	Full Duplex LAN conectada
LNK	Verde	Actividad de Enlace LAN
100	Amarillo	LAN conectada a 100Mbps

1.3.3.1.6 (#7) USB

Las funciones del USB 2.0 de Raspberry Pi están provistas por el chip SMSC LAN9512. El máximo de corriente recomendado es de 100 miliamperios que representa 20% de un USB normal, si se requiere exceder este valor es necesario conectar un HUB-USB que contenga su propia fuente de poder, ya que de no hacerlo se puede dañar la Raspberry Pi o podría no funcionar correctamente. [15]

1.3.3.1.7 (#8)Puerto de Red Ethernet

Se encuentra conectado a otro puerto USB de baja corriente, y posee varias funciones, como la de descarga, que permiten liberar el procesamiento de tareas al CPU principal. [15]

1.3.3.1.8 (#9) Cámara

Su conector posee la interface estándar CSI (Camera Serial Interface), la cual permite a la Raspberry Pi definir el llamado a la interfaz de comandos de control.

La cámara puede de esta manera comunicarse con el GPU directamente, donde la cámara será el esclavo y el GPU el maestro. [15]

1.3.3.1.9 (#10) USB/Ethernet Controler

El chip SMSC LAN9512 es el encargado de controlar los puertos USB y el de Ethernet, ya que se puede notar que tiene un puerto USB de alta corriente sirve para comunicarse con todo el computador y tres de baja corriente, donde uno de los USB de baja corriente se encuentra conectado internamente al puerto de Ethernet. [15]

1.3.3.1.10 (#11) HDMI

Este puerto posee el estándar HDMI 1.3a por lo que únicamente señales de HDMI y DVI-D son soportadas.

Si se tienen conectados los dos puertos de video RCA y HDMI al mismo tiempo, la Raspberry Pi preferirá activar la interfaz HDMI en lugar de la interfaz RCA. [15]

1.3.3.1.11 (#12) Regulador de Voltaje

La Raspberry Pi incluye un regulador de voltaje en su tarjeta, localizado detrás del puerto micro USB y marcado como RG2, al igual que su capacitor de filtrado marcado como C2. [13]

1.3.3.1.12 (#13) Power Micro USB

Dedicado a proveer 5v de corriente directa a la tarjeta desde una fuente externa. Sin embargo la Raspberry Pi trabaja a un voltaje menor de 3.3 V

1.3.3.2 Dispositivos adicionales

1.3.3.2.1 Tarjeta SD

Esta tarjeta almacenará todos los paquetes e información que necesite el sistema operativo, además almacena la información guardada por el usuario.

Se recomienda que la tarjeta SD sea mayor a 4 Giga bytes y que la clase de la tarjeta se la más alta posible ya que ayuda con mayor velocidad para acceder a la información. [13]

1.3.3.2.2 Fuente de Poder

La fuente de poder debe ser de 5 voltios y una corriente de 1 a 2 amperios, dependiendo del número de periféricos que sean conectados.

Esta fuente de poder puede ser conectada a la Raspberry Pi por medio del conector micro USB o a través de los pines del puerto GPIO (5V, GND), sin embargo al usar los pines del puerto GPIO, es recomendable colocar protecciones ya que la tarjeta no las posee al ser alimentada por este medio. [13]

1.3.3.2.3 Cable HDMI

Es necesario un monitor con entrada HDMI o en su defecto, colocar un convertidor de HDMI a VGA.

1.3.3.2.4 Cámara

La cámara de Raspberry Pi es comparable con la del iphone 4 del 2010, sus características se presentan en la Tabla 1.3. Además su uso libera al CPU del procesamiento que debería realizar si se usara una cámara USB.

La cámara cuenta con una cubierta antiestática, ya que se recomienda no tocar los circuitos de la cámara ni el lente para evitar dejar huellas dactilares en él. [13]

Tabla 1.3. Características de la Cámara Raspberry Pi [13]

Característica	**Descripción**
Dimensiones	25mm x 20mm x 9mm
Peso	3g
Tipo de sensor y modelo	Omnivision OV5647
Resolución del sensor	5 megapixeles
Tipo de foco	Ajustado
Ajuste de resolución de la imagen	2592x1944 pixeles
Resoluciones de video	1080 pixeles @ 30 cuadros por segundo; 720 pixeles @ 60 cuadros por segundo; 640x480 @ 60 o 90 cuadros por segundo
Flash	No
Consumo de energía	100 miliamperios a 1.5 voltios

1.3.4 PUERTO GPIO (GENERAL PORPOUSE IN/OUT) - MODELO PI 2

El puerto GPIO varía su número de pines dependiendo del modelo de la tarjeta. Puede tener 26 pines o 40 pines.

Puede observarse cerca al extremo del puerto la etiqueta P1 o P1-01 que indica que el pin contiguo es el primero, el que se encuentre en la fila opuesta será el segundo y así sucesivamente hasta terminar las filas. Existen pines con etiquetas como “NC” o “DNC” que significan “no connect” o “do not connect”

Todos los pines entrada y salida son capaces de generar PWM para control de motores. Los pines 2 y 6 pueden ser usados para alimentar a la Raspberry Pi en lugar de usar el conector micro USB.

Es importante indicar que los pines funcionan con un voltaje máximo de 3.3 voltios y entregan una corriente máxima de 50 miliamperios. [15], [13]

En la Figura 1.27 se presenta la descripción de los pines de la Raspberry Pi 2 Modelo B.

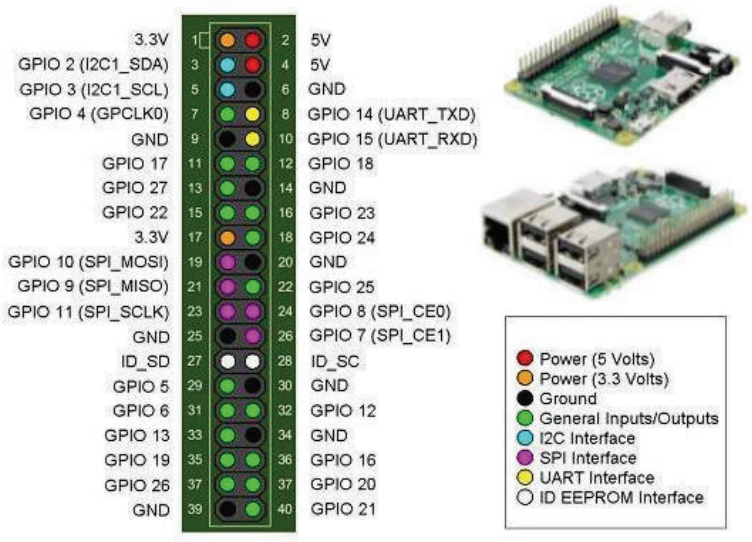


Figura 1.27. Pines del puerto GPIO de la Raspberry Pi 2 Modelo B

1.3.5 CONEXIÓN DE RED EN RASPBERRY PI

1.3.5.1 Red Cableada

Todos los modelos a partir del modelo B de Raspberry Pi soportan este tipo de conexión por medio de su conector RJ-45, a través de un patch cord categorías 5e o 6.

Al conectarse, la Raspberry Pi automáticamente funcionará en la red a 100 Megabytes por segundo (Mbps), con una dirección IP v4 entregada por el router de la red. Los datos de la red pueden visualizarse por medio del comando *ifconfig* escrito en el terminal.

Los leds de estado que se activarán son FDX, LNK y 100. FDX y 100 permanecerán encendidos mientras que LNK parpadeará cada vez que un dato sea enviado o recibido. [13]

1.3.5.2 Red Wireless

Se realiza por medio de un adaptador USB Wifi. Este dispositivo consume mayor corriente, por lo que es necesario aumentar la cantidad de corriente que entrega la fuente de poder.

En este caso los leds de estado que indican la conexión ethernet no se encenderán.

Es necesario realizar la configuración previa de esta conexión y reconocimiento del dispositivo wifi en la Raspberry Pi.

A través de esta conexión se puede acceder remotamente a la Raspberry Pi, por medio de un computador que se encuentre dentro de la red. [13]

1.3.5.3 Protocolo SSH

Es el protocolo usado para acceder remotamente a la Raspberry Pi. Debido a su uso continuo en el trabajo con la Raspberry Pi es indispensable conocer su definición y características.

1.3.5.3.1 SSH (Secure Shell)

Es una aplicación que significa “versión segura del programa Remote Shell”, la misma que define un protocolo para transmisión segura de datos dentro de una red.

El protocolo SSH (Secure Shell) es un protocolo de comunicación segura. Está ubicado debajo de la capa de transporte y permite:

- Establecer conexiones seguras
- Redirección de puertos TCP
- Comunicación entre clientes y servidores de ventanas X [17]

1.3.5.3.2 Características del protocolo SSH

- Confidencialidad: ya que aplica un cifrado simétrico a los datos, haciendo necesario el intercambio de claves entre servidor y cliente.
- Autenticación de entidad: autentica al ordenador servidor como al usuario

- Autenticación de mensaje: se garantiza adicionando un código MAC calculado por medio de una clave secreta.
- Eficiencia: con el fin de reducir la longitud de los paquetes transmitidos los datos son comprimidos.
- Extensibilidad: se utilizan algoritmos para el cifrado, autenticación, compresión e intercambio de claves. [17]

1.4 CONTROLADOR PID

En la actualidad la mayoría de los sistemas industriales son controlados por esquemas PID o PID modificado. La utilidad de los controladores PID se ve reflejada en que su aplicación es casi general para la mayoría de los sistemas de control, en particular cuando no se conoce el modelo de la planta a controlar, por lo tanto no se pueden utilizar métodos de sintonización analíticos. Su facilidad para sintonizarlo en el sitio de trabajo lo concibe como el método más utilizado industrialmente. [18]

1.4.1 ACCIONES DE CONTROL DE UN PID

1.4.1.1 Proporcional

Tiene una implementación discreta directa, multiplica cada muestra de la señal de error por un valor de ganancia Kp ajustable.

Es decir:

$$\mu(t) = Kp * e(t) \tag{1.15}$$

Cuya función de transferencia es:

$$Cp(s) = Kp \tag{1.16}$$

Puede controlar cualquier planta estable, aunque es un controlador limitado y presenta error en estado estable. [19]

1.4.1.2 Integral

Entrega a la salida del controlador un valor proporcional al error acumulado, lo que implica que es un controlador lento. Este controlador asegura un error en régimen permanente igual a cero. [19]

$$\mu(t) = Ki \int_0^t e(\tau)d\tau \tag{1.17}$$

$$Ci(s) = \frac{Ki}{s} \tag{1.18}$$

1.4.1.3 Proporcional-Integral

Está definido por:

$$\mu(t) = Kp * e(t) + \frac{Kp}{Ti} \int_0^t e(\tau)d\tau \tag{1.19}$$

Y su función de transferencia resulta:

$$C_{PI}(s) = Kp\left(1 + \frac{1}{T_i s}\right) \tag{1.20}$$

Dónde:

Ti: Tiempo integral, que se ajusta para la acción integral.

Este control es adecuado para procesos con una función de transferencia de primer orden. El control proporcional necesita un error distinto de cero para realizar su acción, y el control integral realiza su trabajo aunque el error sea pequeño, positivo o negativo. [19]

1.4.1.4 Proporcional-Derivativa

Está definido por:

$$\mu(t) = Kp * e(t) + KpTd \frac{de(t)}{dt} \tag{1.21}$$

Y su función de transferencia resulta:

$$C_{PD}(s) = Kp + sKpTd \tag{1.22}$$

Dónde:

Td: Tiempo derivativo

La acción derivativa nunca se usa sola, debido a su eficacia solo en periodos transitorios.

Esta acción de control es más rápida, aunque tiene la desventaja de amplificar las señales de ruido, dando la posibilidad de saturar el actuador. Al combinar la acción derivativa y proporcional se obtiene un controlador de alta sensibilidad, ya que responde a la velocidad de cambio del error, por lo que puede hacer la acción de control antes de que el error aumente más. [19]

1.4.1.5 Proporcional-Integral-Derivativa

Reúne las ventajas de cada una de las acciones de control anteriores, la ecuación de esta acción se obtiene mediante: [19]

$$\mu(t) = Kp * e(t) + \frac{Kp}{Ti}\int_0^t e(\tau)d\tau + KpTd\frac{de(t)}{dt} \tag{1.23}$$

Y su función de transferencia resulta:

$$C_{PI}(s) = Kp\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{1.24}$$

1.4.1.6 Sintonización del PID por Ziegler and Nichols

1.4.1.6.1 Método de Oscilación

Método propuesto por Ziegler y Nichols en 1942, quienes lo desarrollaron empíricamente. [19]

Procedimiento:

- Variar Kp hasta que el sistema empiece a oscilar. Se requieren oscilaciones lineales que serán observadas a la salida del controlador. [19]
- Registrar la ganancia crítica del controlador Kp=Kc y el periodo de oscilación de la salida del controlador (Pc).
- Ajustar los parámetros del controlador según la Tabla 1.4: [19]

Tabla 1.4. Parámetros de ajuste (método de oscilación) [19]

	Kp	Ti	Td
P	0.50Kc		
PI	0.45Kc	$\frac{Pc}{1,2}$	
PID	0.60Kc	0.5Pc	$\frac{Pc}{8}$

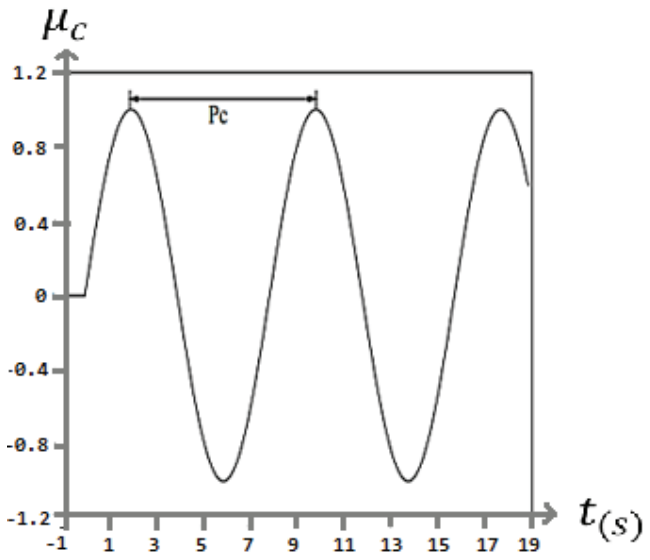


Figura 1.28 Respuesta de la planta con ganancia crítica

1.5 COMUNICACIÓN I2C

La comunicación I2C (Inter-Integrated Circuits), es un sistema de comunicación desarrollado por Philips para comunicar circuitos integrados dentro de un mismo entorno. Esta comunicación ha ido creciendo de manera que se puede controlar en la actualidad una gran variedad de dispositivos.

La comunicación I2C es sincrónica y usa solo 2 líneas de comunicación:

- **SCL (Serial Clock):** La línea del reloj es usada para indicar cuando el dato en la línea SDA es válido.
- **SDA (Serial Data):** Es una línea bidireccional que escribe los datos enviados por el maestro al esclavo y lee los datos desde el esclavo al maestro

Usando solo estas dos líneas es posible enviar información en un sentido a la vez, es decir, de maestro a esclavo, y de esclavo a maestro una vez terminada la comunicación de maestro a esclavo, ya que los datos van hacia el maestro cuando el esclavo está respondiendo un requerimiento del maestro.

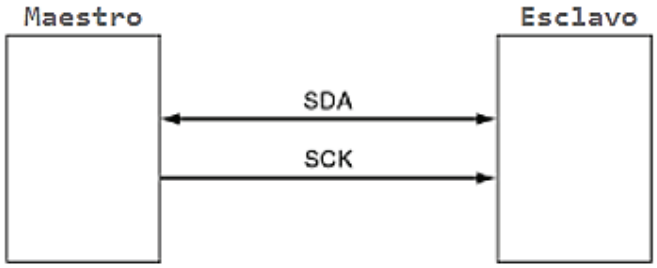


Figura 1.29. Comunicación I2C

El maestro inicializa toda la comunicación enviando la condición de inicio, posteriormente se envía la dirección del dispositivo esclavo al que se desea identificar, el dispositivo esclavo direccionado envía como respuesta un bit ACK para indicar que recibió el comando. La secuencia de intercambio de información se realiza con uno o varios bytes que toman su lugar dependiendo si el tiempo en el que se encuentran es de lectura o escritura.

En el caso de escritura el maestro continúa enviando datos hacia el esclavo, un byte a la vez, con el esclavo enviando un bit ACK en respuesta.

En el caso de lectura el maestro reconoce el bit ACK inicial del esclavo y la dirección del esclavo que fue enviada inicialmente. Y los datos en el reloj y la línea SDA. Cada byte de datos es reconocido por el maestro con un ACK, excepto el último byte el cual es reconocido por un NAK. Toda la comunicación termina cuando el maestro envía un bit de parada al esclavo.

Las tres condiciones que maneja el bus I2C se muestran a continuación:

- **Condición de Inicio:** esta condición precede a toda la comunicación. Se caracteriza porque la línea de SDA cambia de un valor alto a bajo, mientras la línea SCL se encuentra en alto.

- **Condición de parada:** esta condición termina la transferencia I2C. Se caracteriza porque la línea SDA cambia de un valor bajo a alto, mientras la línea SCL se encuentra en alto.
- **Transferencia de datos:** después de la condición de inicio, un bit es considerado valido si se mantiene estable por el tiempo en el que el SCL se encuentre en alto. Cambios en el SDA deberán ser realizados mientras el SCL se encuentre en bajo.

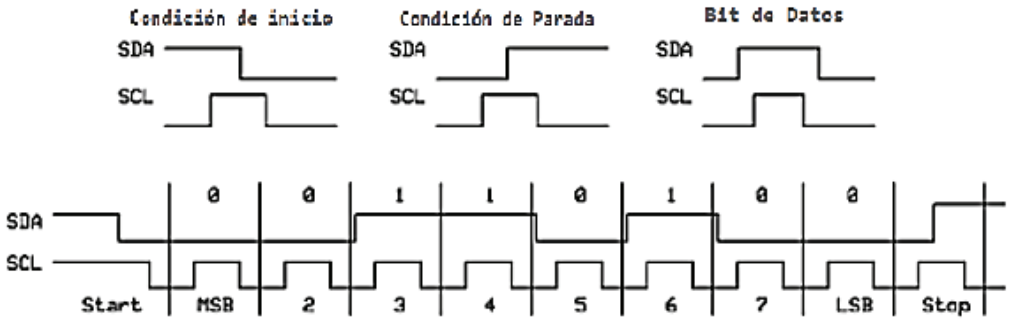


Figura 1.30. Condiciones y envío de datos por I2C

CAPÍTULO 2

DISEÑO Y CONSTRUCCIÓN DEL ROBOT MOVIL

Este capítulo presenta la descripción de los diferentes tipos de robot móviles, entre los cuales se menciona el tipo de robot diseñado en este proyecto. Además las partes que constituyen el robot, su diseño, características y diagrama de conexiones con el cual se muestra la sinergia de todas las partes.

2.1 TIPOS DE ROBOTS MOVILES

2.1.1 ROBOTS MÓVILES TERRESTRES

Existen varios tipos de robots móviles terrestres dependiendo del tipo de locomoción que se tenga, de esto difiere las ventajas y desventajas de cada uno de ellos, las cuales dependerán directamente del tipo de terreno sobre el cual se podrán movilizar, entres los tipos más comunes de robot móviles se tiene lo siguiente:

2.1.1.1 Tipo Oruga

En estos tipos de robots se realiza una sola tracción, es decir, unen la rueda trasera con la delantera por medio de un acople, que puede ser de goma o metálico, de esta manera pueden movilizarse en terrenos irregulares o terrenos más complejos gracias a su gran área de contacto, la desventaja de este tipo de robots es el movimiento de trayectoria que se tiene, ya que solo pueden moverse en línea recta, no pueden realizar trayectorias curvas. [20]

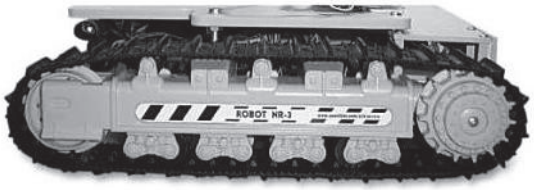


Figura 2.1. Robot tipo Oruga [20]

2.1.1.2 Patas

Estos tipos de robots implementan un sistema de movilización por medio de patas, las cuales transitan por terrenos más complejos. Entre más patas tenga el

robot mayor es el equilibrio obtenido. En general presentan tres grados de libertad como es el caso de un robot humanoide (cadera, rodilla, y tobillo). [20]

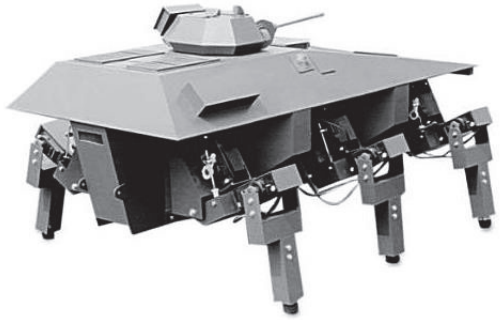


Figura 2.2. Robot con patas [20]

2.1.1.3 Ruedas

Estos tipos de robots implementan en su sistema de movilización ruedas, lo cual es muy beneficioso para terrenos planos sin irregularidades, la potencia del robot depende del número de ruedas, pueden ser de tipo tracción diferencial, uniciclo (dos ruedas controladas y una rueda directriz), tipo Akerman (Cuatro ruedas controladas), sincronizado, omnidireccionales.

Los robots con ruedas son más sencillos y con menos complicaciones en la construcción, la desventaja es su movilización en terrenos irregulares.

2.1.1.3.1 Tracción Diferencial

Este sistema de locomoción, permite moverse al robot en forma recta, girar sobre su propio eje y trazar curvas. El mayor inconveniente en estos tipo de locomoción es el equilibrio del robot debido a que necesita apoyo adicional ya sea triangular (un apoyo) o romboidal (dos apoyos), esta consideración se la debe hacer dependiendo del tipo de carga que se tenga. El romboidal puede soportar más peso que el triangular. Otro inconveniente es hacer que el motor se mueva en forma recta ya que las ruedas deben tener la misma velocidad, lo que no sucede debido a que los motores vienen dados con diferente resistencia en su bobinado, lo que ocasiona que al mismo voltaje aplicado en las dos ruedas, estas no se muevan a la misma velocidad, para solucionar este inconveniente las ruedas deben ser controladas de forma dinámica, es decir, debe existir un medio para monitorizar y cambiar la velocidad mientras el robot este avanzando, por lo tanto el control de las ruedas resulta ser complejo a comparación del diseño. A pesar

de eso en relación a costos este tipo de locomoción resulta ser el más económico. [21]

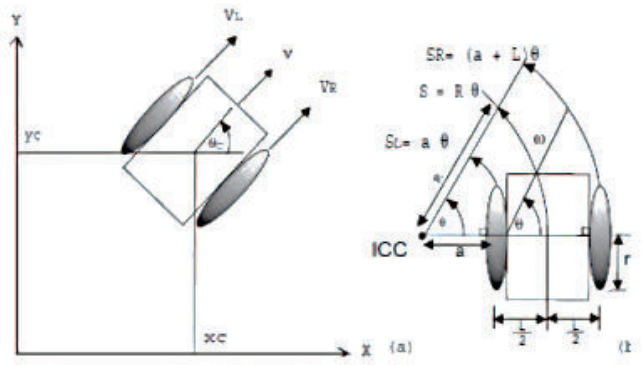


Figura 2.3. Ecuaciones del modelo cinemático configuración diferencial [22]

2.1.1.3.2 Sincronizado

Este tipo de diseño consta de 3 o más ruedas, mismas que son a la ves directrices y motrices, se encuentran acopladas a un cuerpo recto y en la misma dirección. Para que el robot pueda girar, las ruedas se mueven simultáneamente alrededor de un eje vertical pero el chasis sigue apuntando a la misma dirección. Este tipo de diseño superó muchas dificultades que han tenido los otros robots de tracción diferencial, pero con mucha complejidad en el diseño mecánico. [21]

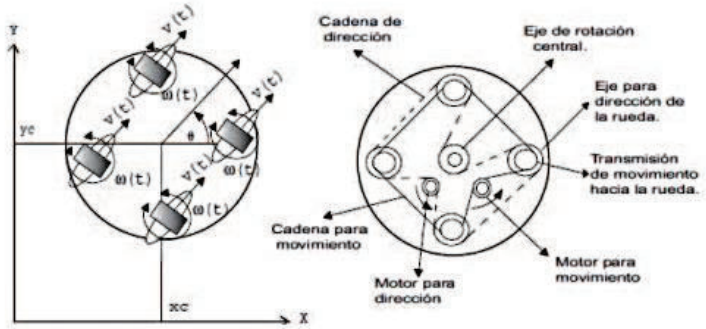
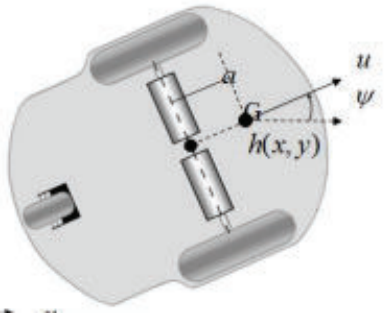


Figura 2.4. Modelo cinemático, configuración sincronizada [22]

2.1.1.3.3 Uniciclo

El diseño de tipo uniciclo consta de dos ruedas con motores acoplados y una rueda loca que beneficia el libre movimiento del robot. El control para los motores puede ser de tracción diferencial con el monitoreo continuo de las velocidades, estos tipos de robots son estables y pueden soportar peso encima dependiendo del material de construcción del chasis y la potencia de las ruedas.



u=velocidad lineal

$\psi = ángulo$

G= centro gravedad

h(x,y)=posición

Figura 2.5. Robot tipo uniciclo [23]

2.1.1.3.4 Akerman

El diseño de tipo Akerman consta de cuatro ruedas siendo dos de ellas directrices y las otras dos motrices estáticas. Presentan dos ángulos de giro en las ruedas directrices si se trabajan independientemente, por lo cual presentan problemas para realizar el control, para solucionar estos problemas se unifican los ángulos de giro por medio de una barra para facilitar el control. Estos tipos de robots son similares los carros convencionales de pasajeros.

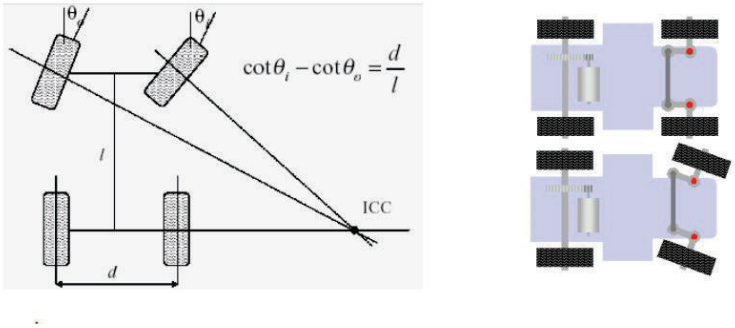


Figura 2.6. Robot tipo uniciclo

2.1.1.4 Ápodos

Este tipo de robots no emplean en el sistema de movilización ruedas, oruga o patas, el modelo a seguir es el de las serpientes o gusanos los cuales tienen partes articuladas en forma modular, pueden moverse en varios tipos de terrenos ya sea irregulares o regulares, la forma de movimiento es lineal, logrado por un efecto de onda formado desde la cola hacia la parte delantera del robot. [20]

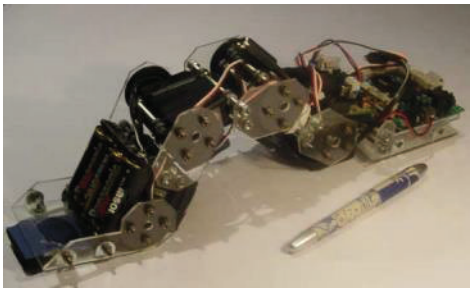


Figura 2.7. Robot Apodo [24]

2.2 DESCRIPCIÓN DEL HARDWARE DEL ROBOT

2.2.1 MOTORES

Los motores usados para el diseño son los micro motores marca Pololu. Estos pueden trabajar sin inconvenientes en el rango de 3v a 9v. La rotación de los motores empieza en valores tan bajos como 0.5v, menores voltajes no serían prácticos, y mayores voltajes pueden afectar negativamente la vida útil del motor [25] . En la Tabla 2.1 se muestran las características del motor Pololu.

Figura 2.8. Micro motor Pololu 210:1, [25]

Tabla 2.1. Dimensiones y características generales del motor pololu. [25]

Dimensiones	
Medidas	10 x 12 x 26 mm
Peso	9.5 g
Diámetro del eje	3 mm
Especificaciones Generales	
Radio de Engranes	210.59:1
Velocidad en libre movimiento	140 rpm
Corriente en libre movimiento	120 mA
Corriente con rotor bloqueado	1600 mA
Torque con rotor bloqueado	50 oz-in
Larga vida de escobillas de carbón	Si
Tipo de Motor	1.6 A en paro @ 6v (HPCB-escobillas de carbón)

2.2.2 DRIVER PARA MOTOR L298D

El driver para motor L298, es un módulo basado en el chip L298N, que permite controlar motores de corriente continua o motores paso a paso bipolares con una corriente de hasta 2 amperios la cual logra abastecer sin problemas la corriente que demandan los motores.

Cuenta con un regulador LM7805 el cual se encarga de suministrar el voltaje al chip L298N, posee jumpers de selección para habilitar cada una de las salidas del módulo (A y B).

La salida A está conformada por OUT1 y OUT2, y la salida B por OUT3 y OUT4. Los pines de habilitación son ENA y ENB respectivamente.

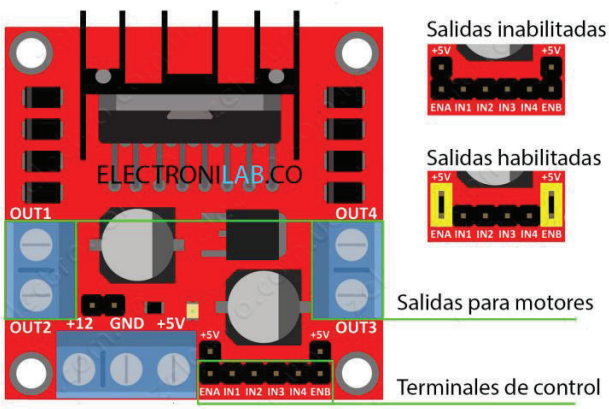


Figura 2.9. Diagrama de Entradas, salidas y terminales del Driver L298D

2.2.3 PLATAFORMA COMERCIAL DE 32 BITS (SBC)

La plataforma comercial de 32 bits utilizada en este proyecto es la Raspberry Pi 2 Modelo B, debido a que es económica y las características técnicas son suficientes para la realización de este proyecto, cuya descripción más detallada se encuentra en el apartado 1.3 RASPBERRY PI.

La información referente a esta tarjeta, creada por la fundación Raspberry Pi, es amplia y detallada por personas de todo el mundo y más aún en la página oficial de Raspberry Pi. Al trabajar bajo software libre proporciona a su usuario mayor ayuda para la resolución de problemas que pueden presentarse como: compatibilidad de versiones entre programas, instalación de paquetes, actualizaciones, configuración de archivos del sistema, etc.

Una de sus mayores ventajas es su consumo, ya que en reposo únicamente consume 0.5W y al ejecutar un programa la demanda de energía aumenta y puede llegar al rango entre 3.5W-4.4W. Con esto se convierte en el SBC idóneo para la creación de proyectos móviles, ya que su independencia por medio de alimentación por baterías será mayor.

Su uso en este proyecto de visión por computador, representa el 100% de la capacidad de la Raspberry Pi, la cual permite realizar el procesamiento de imágenes con un máximo de 10 frames por segundo. Sin embargo al realizar generación de señales para el control de motores, envío de señales por I2C a otro microprocesador, entre otras funciones, la velocidad de procesamiento disminuye.

Una de las desventajas de la Raspberry Pi es que al necesitar usar el 100% de su capacidad, se puede producir recalentamiento ya que no se cuenta con un sistema integrado de refrigeración. [26]

2.2.4 CÁMARA

La cámara utilizada es la Logitech C170, cuyas características y requerimientos se presentan en la Tabla 2.2 y

Figura 2.10 Partes de la Cámara Logitech C170 [22]

Tabla **2.3** respectivamente. La cámara tiene la característica de ser Plug and Play por lo que al conectarla en la Raspberry Pi es reconocida automáticamente, cualquier otra cámara debe tener previamente instalado su driver, lo que consume espacio en la Raspberry Pi.

Tabla 2.2. Especificaciones de la Cámara Logitech C170 [27]

Especificaciones	Descripción
Peso	290 g
Longitud	20.95 cm
Ancho	7.62
Alto	15.24
Video Calling	640 x 480 pixeles
Video Capture	1024 x 768 pixeles
Fotos	Hasta 5 megapixeles
Micrófono	Integrado, con reducción de ruido
USB	Versión 2.0 de alta velocidad

Tabla 2.3. Requerimientos del sistema que utilice la cámara Logitech C170 [27]

Requerimientos del Sistema	Descripción
Procesador	1 GHz (1,6 GHz recomendado)
Memoria	500 MB RAM o más
Espacio en disco	200 MB
Conexión a internet	Requerida
USB	Versión 1.1 (2.0 recomendado)

2.2.5 ADAPTADOR WIRELESS USB

Utilizado en este proyecto para poder comunicar a la Raspberry Pi inalámbricamente con cualquier pc dentro de la red, para así mantener la autonomía del robot. Este adaptador Wireless presenta la ventaja de que consume poca energía, lo que aumenta el tiempo de funcionamiento del robot.

Figura 2.11 Adaptador Wireless USB para Raspberry Pi

2.2.6 ROUTER

Se usó el router TP-Link por ser una marca comercial de gran compatibilidad con la mayoría de sistemas.

2.2.6.1 Panel frontal

La descripción de los indicadores visuales del panel frontal se muestra en la Tabla 2.4

Tabla 2.4. Panel frontal router TP-LINK [28]

Nombre	Estado	Descripción
Fuente	Apagado	Fuente apagada
	Encendido	Fuente encendida
Sistema	Apagado	El router tiene un error de sistema
	Encendido	1. El router esta iniciando 2. El trabajo del router tiene un error de sistema
	Parpadeante	El router esta funcionando apropiadamente
WLAN	Apagado	La función inalámbrica esta deshabilitada
	Parpadeante	La función inalámbrica esta trabajando apropiadamente
LAN WAN	Apagado	Ningún dispositivo esta conectado a su correspondiente puerto
	Encendido	Hay dispositivos conectados en el correspondiente puerto pero sin actividad.
	Parpadeante	Hay dispositivos conectados en el correspondiente puerto y se encuentran activos.

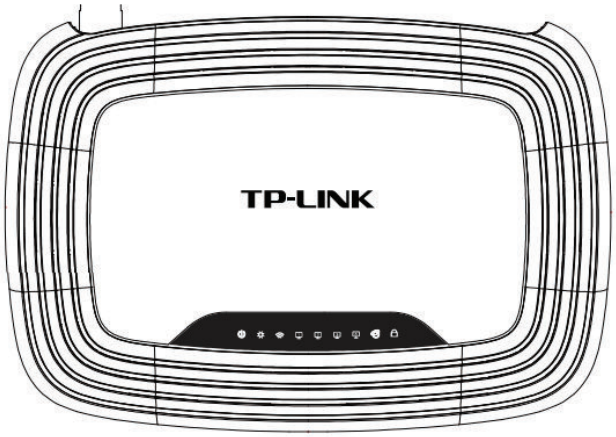


Figura 2.12. Panel Frontal [28]

2.2.6.2 Parte posterior

Está conformado por las siguientes partes:

- Socket para la conexión de la fuente de poder. [28]
- Cuatro puertos de conexión LAN para conectar el router a un PC. [28]
- Un puerto para conexión WAN, donde se debe conectar el cable del modem o ethernet. [28]
- Botón de reset, el cual funcionara si es presionado por más de 5 segundos. [28]

Figura 2.13. Parte posterior del router TP-LINK [28]

2.2.6.3 Características Principales

- Cumple con la norma IEEE 802.11n para proveer una velocidad de datos inalámbricos de hasta 150Mbps. [28]
- Un puerto de conexión WAN RJ45, cuatro puertos de conexión LAN RJ45. [28]
- Encriptación de seguridad TKIP/AES y autentificación WPA/WPA2, WPA-PSK/WPA2-PSK. [28]
- Acceso para usuarios a datos compartidos e internet, soporta IP estáticas y dinámicas. [28]
- Soporta UPnP, DNS dinámico, enrutamiento estático. [28]
- Soporta conexión de internet IPV6. [28]
- Soporta flujos estáticos. [28]

2.2.7 FUENTE DE ENERGÍA

La fuente de alimentación está dividida en dos: fuente de los motores y fuente del control.

2.2.7.1 Fuente del control

La fuente de energía para el control (conformado por la Raspberry Pi, todos sus periféricos conectados y la cámara), es el Power Bank Marca: Samsung, Modelo: LO11. Cuyas características se muestran en la Tabla 2.5

Las ventajas de usar este dispositivo como alimentación de energía son:

- Muestra la cantidad de carga disponible en la batería, por medio de sus leds.
- Tiene dos salidas de 1A y 2A.
- Carga por medio de un conector micro USB.
- Apagado automático si no se registra ninguna carga.

Figura 2.14. Power Bank Samsung. [29]

Tabla 2.5. Características del Power Bank Samsung

Características	**Descripción**
Marca	Samsung
Modelo	LO11
Capacidad	9000 mAh
Voltaje de Entrada	5 V
Voltaje de Salida	5.5 V
Corriente de Entrada	1.8 A
Corriente de Salida	Salida 1: 2 Amp., Salida 2: 1 Amp.
Leds indicadores de carga	Si

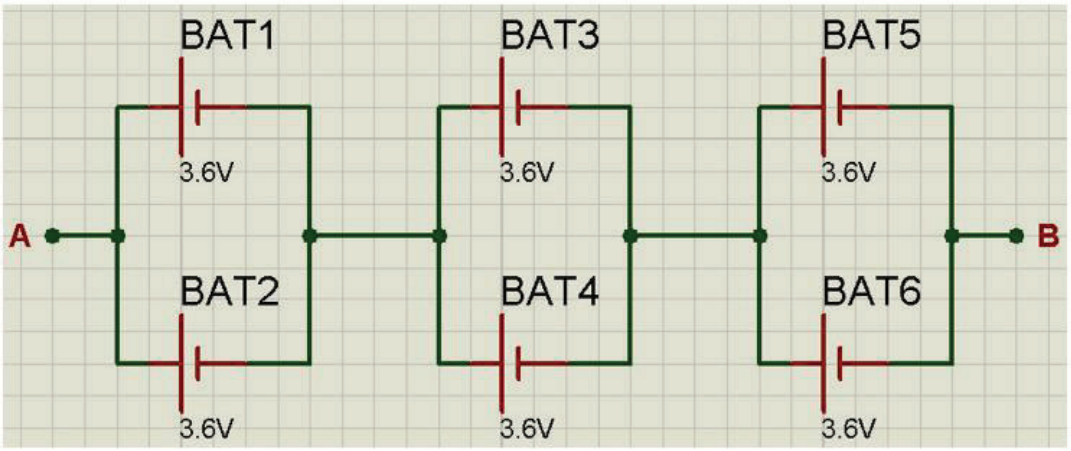
2.2.7.2 Fuente de los motores

La fuente de los motores está formada por dos baterías de 11V y 5225mAh. Las cuales se encuentran conectadas de acuerdo al diagrama de la Figura 2.15.De donde se puede calcular:

$$V_{BATERIA} = \frac{11\ V}{3} = 3.66\ V \tag{2.1}$$

$$I_{BATERIA} = \frac{5225\ mAh}{2} = 2612\ mAh = 2.6\ Ah \tag{2.2}$$

a)



b)

Figura 2.15 Banco de baterías a) Batería Física laptop HP b) Diagrama de conexión

Una vez obtenido el voltaje y corriente de cada batería y de acuerdo a la demanda de voltaje y corriente de los motores descritos en la Tabla 2.1, se procedió a implementar la fuente de los motores de acuerdo al diagrama de la Figura 2.16. Con esta conexión se logra obtener un voltaje de 7.2 V y una corriente de 2.6 Ah, suficientes para cubrir la demanda de voltaje y corriente de los motores.

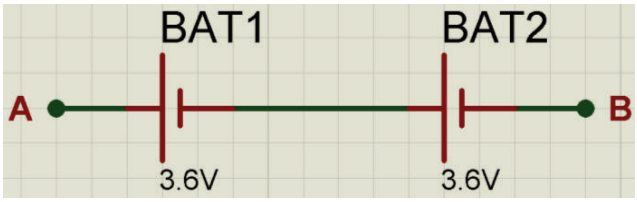


Figura 2.16. Fuente de energía para los motores

2.3 DISEÑO DE LA ESTRUCTURA DEL ROBOT MÓVIL

Con la finalidad de obtener un lugar para alojar a los dispositivos comprendidos en el sistema de visión artificial tales como: motores, cámara, drivers para motores, llantas, tarjeta procesadora de control “Raspberry pi”, etc. Se realizó el diseño de un modelo típico de automóvil.

Para realizar este diseño estructural se acudió a un programa llamado INVENTOR versión 2014, el cual nos permite dibujar cada una de las piezas que intervienen en el sistema, para luego realizar la estructura procurando que todas las piezas compaginen en la misma.

Se usó ruedas comerciales las cuales se encuentran fabricadas de plástico común y caucho que fácilmente se adhieren a los suelos planos sin irregularidades; para dibujarlo se tomó en cuenta todos los detalles y medidas necesarias para que el dibujo se asemeje a la realidad obteniendo el siguiente resultado.

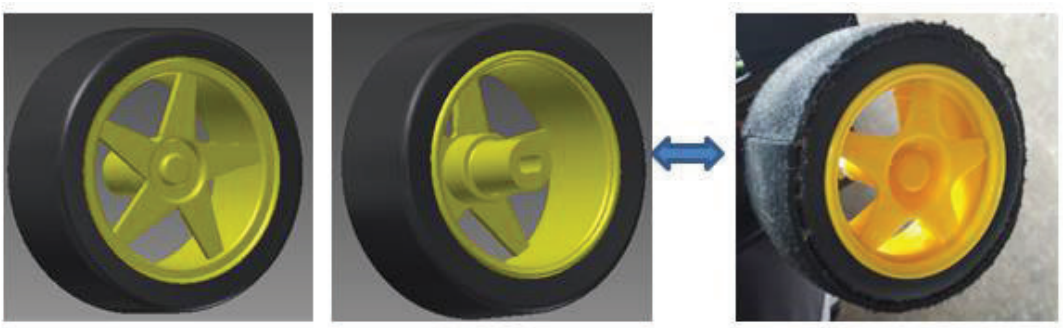


Figura 2.17. Comparación entre Rueda Real (Derecha) y su diseño en Inventor (Izquierda)

Se usó micromotores de fabricación POLOLU los cuales se encuentran fabricados de metal y contienen una caja reductora, lo que hace que sea más potente a comparación de otros motores DC, obteniendo el siguiente dibujo.

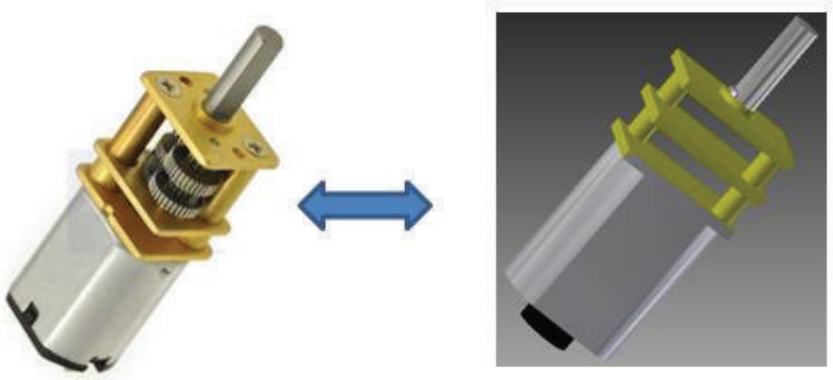


Figura 2.18. Comparación entre Motor Pololu (Izquierda) y su diseño en inventor (Derecha).

Se usó un driver comercial para poder controlar los micromotores, el cual contiene un circuito integrado L298N que puede manejar dos motores, cada uno en ambos sentidos, obteniendo el siguiente dibujo.

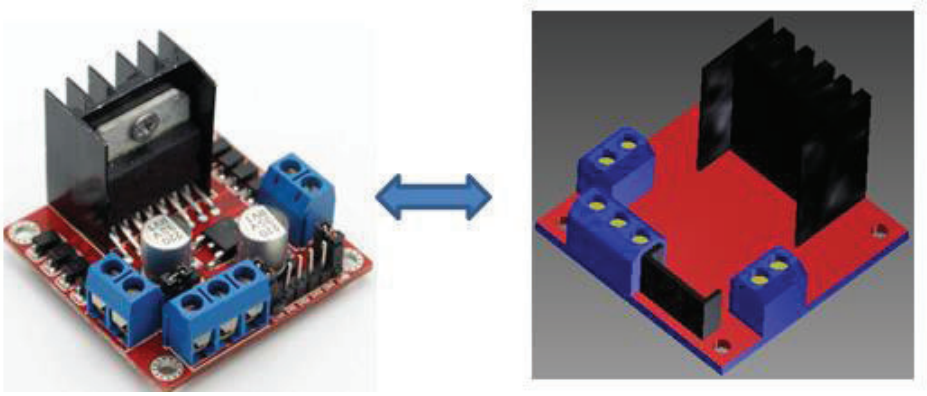


Figura 2.19. Comparación entre el Driver L298N (Izquierda) y su diseño en inventor (Derecha)

Se adquirió una Raspberry Pi 2 modelo B comercial cuyo sistema es base de todo el procesamiento de imagen y cerebro de todo el proyecto, obteniendo el siguiente dibujo.

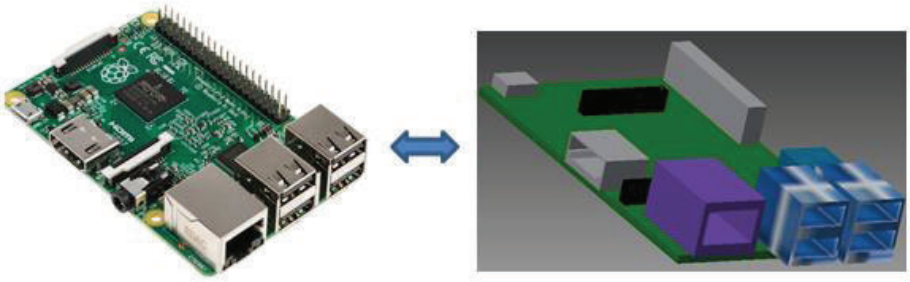


Figura 2.20. Comparación entre Raspberry Pi 2 modelo B (Izquierda) y su diseño en Inventor (Derecha)

Se adquirió una webcam Logitech C170 de conexión USB que es compatible con la Raspberry Pi además puede capturar imágenes 1024x768 pixeles, obteniendo el siguiente dibujo.

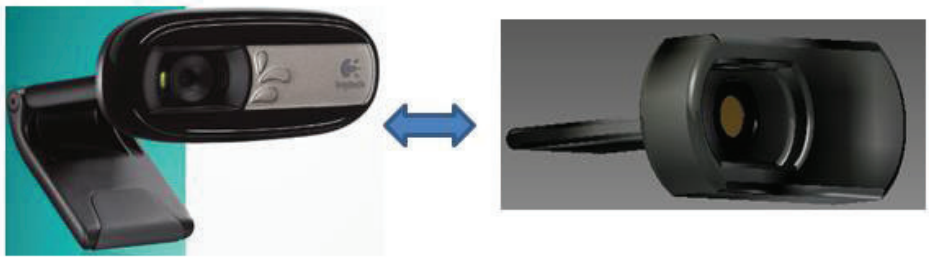


Figura 2.21. Comparación entre webcam Logitech C170 (Izquierda) y su diseño en Inventor (Derecha)

Una vez dibujadas las piezas que componen el robot se diseña la estructura, considerando las medidas y ubicación de todos los elementos, utilizando la creatividad para la optimización de recursos.

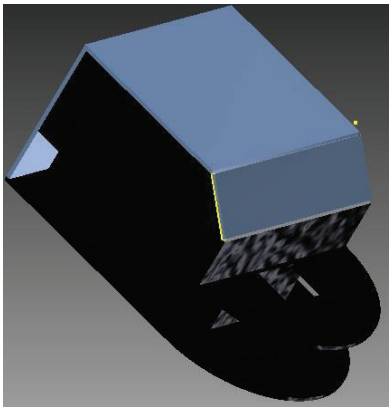


Figura 2.22. Estructura del robot diseñada en Inventor

Una vez diseñada la estructura se debe comprobar que todas las piezas encajen en sus respectivos lugares, para obtener una visión general del robot antes de fabricarlo.

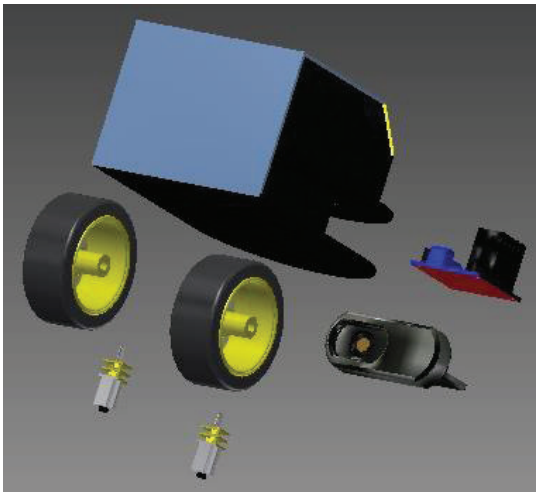


Figura 2.23. Ensamblaje de elementos en la estructura

En las siguientes figuras se puede visualizar la ubicación de los elementos que conforman el robot móvil.

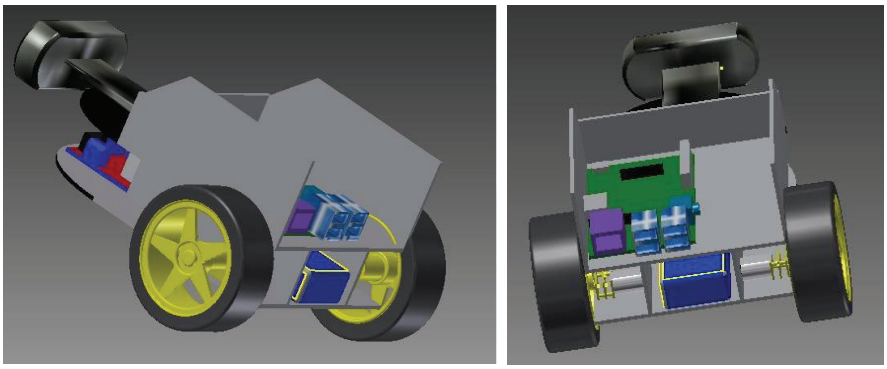


Figura 2.24. Colocación de los elementos en la estructura del robot

Finalmente se observa al robot completo junto con su tapa trasera y superior, obteniendo una perspectiva más real del robot a implementar

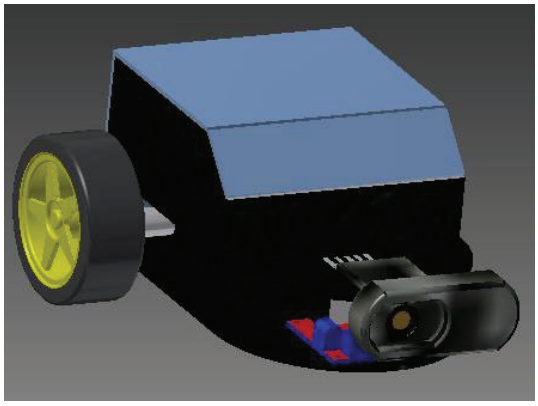


Figura 2.25. Robot con su perspectiva real de implementación

Al tener la estructura dibujada y verificar que todas las piezas estén en el lugar correcto y no haya sobredimensión, el dibujo de la estructura pasa a un plano 2D o directamente a AutoCad, cuyo resultado se observará en la Figura C.1, para proceder a la fabricación.

Una vez fabricada la estructura se adecuan las piezas en su lugar, además se coloca un sistema de sujeción basado en pernos y tuercas para que las piezas se inmovilicen al momento de realizar el seguimiento de objetos utilizando visión artificial.

En la fabricación de la estructura se utilizó acrílico ya que es un material resistente, liviano, fácil de moldear y de buena presentación. Las características más relevantes que se deben tomar en cuenta al momento de escoger un material de fabricación son que sea liviano y resistente a la vez, para no tener inconveniente al momento de colocar las piezas que conforman el robot.

Figura 2.26. Robot implementado

2.4 DIAGRAMA DE CONEXIÓN

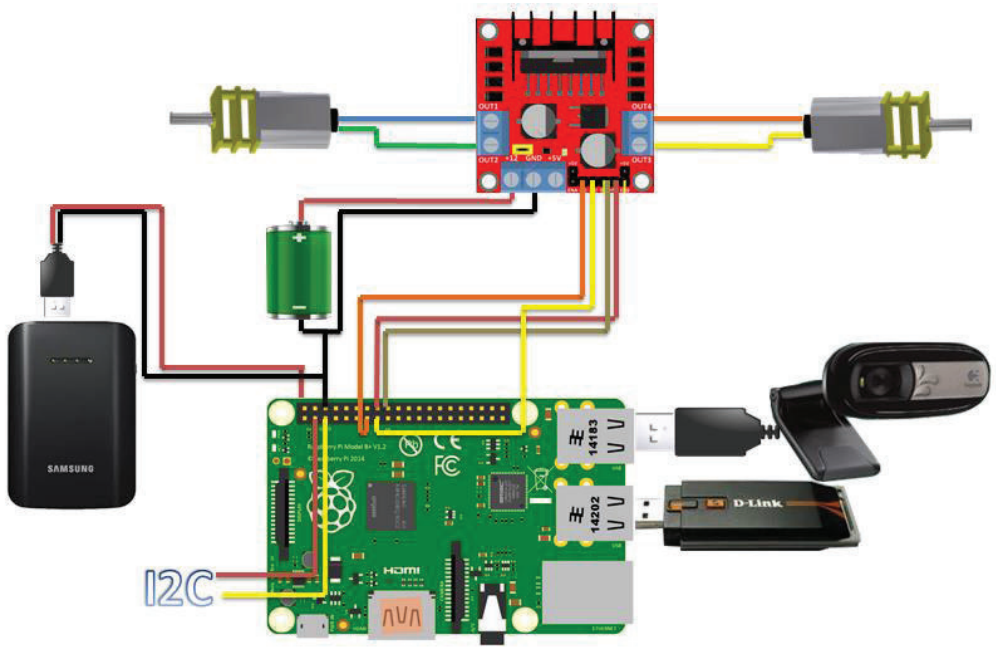


Figura 2.27. Diagrama de conexión de todos los dispositivos del robot

2.5 MODELADO DEL ROBOT MÓVIL

El robot escogido para el desarrollo de este proyecto es del tipo uniciclo, ya que presenta ventajas como movilidad, simple configuración etc.

Existen dos tipos de modelados el cinemático y el dinámico. En este proyecto se verá de manera general el modelo cinemático ya que el dinámico tiene parámetros como peso de cada componente, modelo de los motores, inercia, fricción del suelo etc., es decir, parámetros que son medibles solamente en laboratorios y no es parte del objetivo del proyecto.

2.5.1 MODELO CINEMÁTICO

El modelo cinemático se basa en el estudio del movimiento del robot en función su geometría, este tipo de modelo se usa para el diseño del controlador, incluyendo la simulación del comportamiento cinemático del mismo.

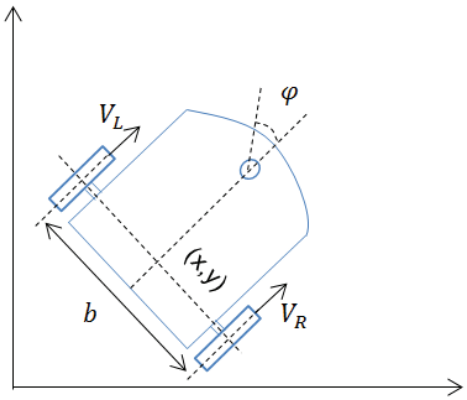


Figura 2.28 Modelado robot de tracción diferencial

Como se puede observar en la Figura 2.28 el robot es de tracción diferencial, las velocidades de cada ruedas son independientes (V_L, V_R), el cual se mueve en el plano 2D (x, y) con orientación φ.

Conociendo que w_l, w_R son las velocidades angulares de cada rueda, r es el radio de las ruedas, b es la distancia entre ruedas y v_l, v_R las velocidades lineales de correspondiente a las ruedas se tienen las siguientes ecuaciones: [30]

$$v = \frac{v_L + v_R}{2} = \frac{w_L + w_R}{2} \cdot r \tag{2.3}$$

$$w = \frac{v_L + v_R}{b} = \frac{w_L + w_R}{b} \cdot r \tag{2.4}$$

El modelo cinemático del robot no holonómico es el siguiente:

$$\dot{x} = vcos(\varphi) \tag{2.5}$$

$$\dot{y} = vsen(\varphi) \tag{2.6}$$

$$\dot{\varphi} = w \tag{2.7}$$

Donde los estados son x, y, φ y las entradas son v, w que son velocidades lineal y angular respectivamente. [31]

En este proyecto no se utiliza el modelo del robot, ya que el ángulo se lo determina de una manera diferente mediante el uso de una cámara digital, lo cual se explicará en el siguiente capítulo.

2.6 CONCLUSIÓN DEL CAPÍTULO

Se logró desarrollar un robot tipo uniciclo, el cual posee una cámara USB como único sensor, utiliza un tarjeta con un sistema embebido para el procesamiento de las imágenes y está provisto de un arreglo de baterías que le dan independencia de funcionamiento por un periodo de aproximadamente 4 horas.

CAPÍTULO 3

DESARROLLO DEL SOFTWARE DE CONTROL

En este capítulo se muestra como se desarrolló el algoritmo de visión por computador y control de todos los periféricos. Presentando los comandos usados de la librería OpenCv, la descripción de sus parámetros y el fundamento matemático que contienen.

Se presentan además los diagramas de flujo del programa, mismos que exponen en forma ordenada el algoritmo utilizado.

3.1 VISIÓN POR COMPUTADORA

Para desarrollar el sistema de control se utilizó una herramienta poderosa llamada OPENCV. En el capítulo uno Figura 1.8 se explicó acerca del modelo de visión el cual se encuentra contemplado en cuatros partes fundamentales que son: captura, preproceso, segmentación y reconocimiento. Para cada parte OpenCV ya cuenta con comandos adecuados el cual se explicará con mayor detalle.

Para poder comenzar a trabajar con opencv se deben de añadir las siguiente librería en Python **import cv2.cv as cv**, **import cv2 as cv2**.

3.1.1 CAPTURA

Una imagen digital es un conjunto de pixeles el cual tienen información importante dependiendo de los canales que se tenga, por ejemplo, si la imagen está en escala de grises la información del pixel es el brillo, dado en la siguiente función:

$$I\,(x, y) \tag{3.1}$$

Donde x y y representa las coordenadas del pixel y la función I representa el nivel del brillo.

En una imagen a escala de grises la profundidad de bits típicamente varía de 2 a 8 bits, si la imagen es de 8 bits se puede representar $2^8 = 256$ tipos de tonalidades de grises. [32]

Si la imagen vienen en tres canales (color), esta información puede venir en RGB dando cuatro matrices: una de posiciones y las restantes tres a características de

cada canal, la profundidad de bits varía entre 8 a 24 bits. Si es de 24 bits generalmente se dividen 8 bits para color verde, 8 bits para el rojo y 8 bits para el azul. Para representar los otros colores se utilizan combinaciones de esos bits. Una imagen de 24 bits ofrece $2^{24} = 16.7\ millones$ de valores de color. [32]

Las imágenes digitales son señales discretas obtenidas de una señal continua, el proceso de obtención de una imagen contiene dos partes, la primera de captura y la segunda de digitalización, estas dos partes son realizadas por un dispositivo óptico, generalmente una cámara que permite obtener y discretizar a la imagen. [33]

3.1.1.1 Apunte al directorio de la cámara

La cámara que ayuda con la captura y digitalización es una Logitech C170 con comunicación USB el cual permite recoger fotografías con una capacidad de 640x480 pixeles. La raspberry pi reconoce directamente el dispositivo y lo coloca en un directorio llamado “Video 0”, en el cual se debe ingresar para la obtención de información usando el siguiente comando:

cv2.VideoCapture(filename-device)

Dentro del paréntesis se debe de colocar el directorio, si es obtenida de una cámara USB este valor varía de [0 a 1] en este caso es “0”. Este comando también funciona para acceder a videos o secuencia de imágenes ya guardados previamente, colocando dentro del paréntesis la dirección donde se encuentre guardado el mismo, los formatos recomendados para video es avi y la secuencia de imágenes jpg. [34]

3.1.1.2 Reducción de la resolución de la imagen

El cambio del tamaño de la imagen corresponde a la matriz de escalado, donde (x, y) es el punto del pixel a ser escalado en un factor (s_x, s_y) obteniendo el nuevo punto (x', y').

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.2}$$

La raspberry pi no es capaz de procesar rápidamente la imagen en la resolución original (640 x 480 pixeles), por lo que se procede a reducir la cantidad de pixeles a un tamaño adecuado para que el proceso sea rápido, y la secuencia de imágenes se observe como un video normal, para lo cual se utiliza el siguiente comando de opencv:

capture.set(cv2.cv.CV_CAP_PROP_FRAME_WIDTH, width)

capture.set(cv2.cv.CV_CAP_PROP_FRAME_HEIGHT, height)

Donde width y height es el tamaño de pixeles en alto y ancho. [34]

3.1.1.2.1 Relación de Aspecto

La relación de aspecto (Ra) es una proporción de tamaños entre el alto y el ancho en relación a los pixeles, cuya fórmula es la siguiente:

$$Ra = \frac{pixeles\ Ancho}{Pixeles\ de\ Alto} \tag{3.3}$$

Existen relaciones de aspecto predefinidas como son 4:3, 3:2, 16:9, 11:9, 1,85:1, 2,38:1, siendo el ancho mayor al alto, basándose en pruebas se determinó las relaciones de aspecto que permiten bajar la resolución en el comando de opencv los cuales se muestran en la siguiente tabla:

Tabla 3.1. Valores predefinidos de resolución

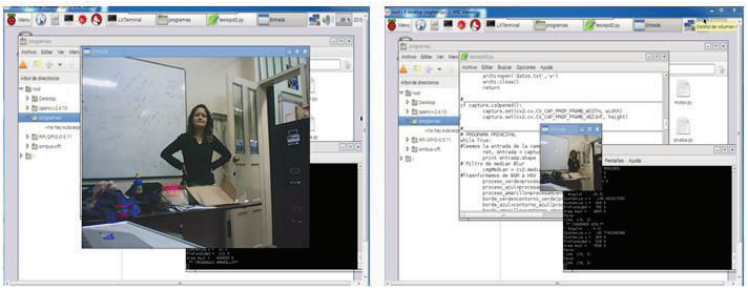
Resolución	**Relación de aspecto**
640 x 480	4:3
320 x 240	4:3
176 x 144	11:9
160 x 120	4:3

Se realizó pruebas para la determinación de la resolución adecuada tanto que el robot pueda reconocer el objeto hasta 4 metros y el proceso sea rápido, los valores de altura y anchura en pixeles de este proyecto es el siguiente:

Width = 176

Height = 144

La cámara con la resolución reducida puede captar 10 frames/segundo.



a) b)

Figura 3.1. a) Imagen resolución 680x480 b) Imagen resolución 176x144

3.1.1.3 Adquisición de la imagen

Una vez dirigido al directorio se debe leer la secuencia de imágenes, utilizando el comando más conveniente para la lectura de datos codificados:

cv2.VideoCapture.read( )→ retval, image.

Este comando adquiere la imagen del directorio donde se encuentra la cámara y retorna un valor booleano (True/False) el cual si la imagen se ha leído correctamente devuelve un True, si no existe imagen ya sea por la desconexión de la cámara o alguna otra razón la misma devuelve un falso y la función retorna un puntero nulo. [34], [9]

3.1.2 PREPROCESO BASADO EN LA IMAGEN

En la imagen digitalizada siempre existen ruidos y perturbaciones los cuales deben manejar filtros que permitan limpiar la imagen de tal manera que en la segmentación no se tenga mayores errores de ruido.

3.1.2.1 Filtro de media

El filtro de media es de tipo espacial lineal el cual se utiliza para el suavizado de una imagen en detalles, consiste en calcular la mediana del conjunto de pixeles vecinos al pixel central, cuyo valor es reemplazado por el valor del cálculo efectuado.

El cálculo de la mediana de los vecinos dependerá del tamaño del kernel, es decir si se tiene una matriz kernel de 3x3 el cálculo de la mediana se realizará a los 8

vecinos del pixel central, la operación matemática que se efectúa a la imagen es la de convolución en dos dimensiones el cual tiene la siguiente ecuación:

$$f(x,y) * g(x,y) = \iint_{\tau_1\tau_2-\infty}^{+\infty} f(\tau_1,\tau_2).g(x-\tau_1,y-\tau_2)\,d\tau_1 d\tau_2 \qquad (3.4)$$

Función discreta

$$f|x,y| * g|x,y| = \sum_{n1}\sum_{n2} f|n1,n2| * g|x-n1,y-n2| \qquad (3.5)$$

Donde $f(x,y)$ es una función desconocida de la imagen donde se realizará la convolución y $g(x,y)$ es una función conocida (kernel plano) dado por el usuario, con los cuales se tendrán que realizar una multiplicación punto a punto, para luego sumarlas en filas y columnas según la ecuación (3.5).

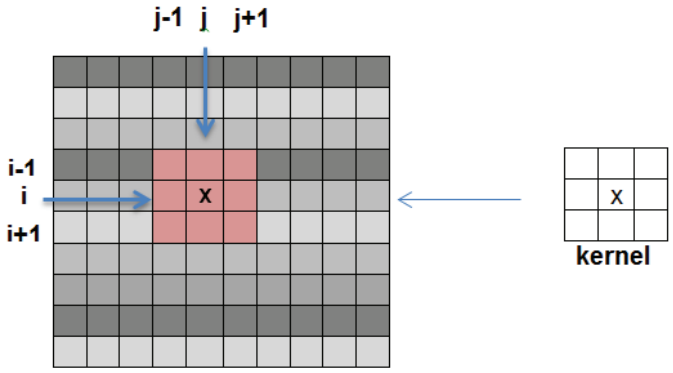


Figura 3.2. Producto punto a punto del elemento de la imagen con el elemento del kernel

Al momento de realizar el filtro de mediana a la imagen, esta se suaviza y pierde los detalles, es decir se elimina todas las frecuencias altas quedando solamente las frecuencias bajas. La aplicación más conocida es la eliminación de ruidos planos o conocidos como sal y pimienta. El procesamiento de este filtro es bastante intensivo dependiendo de la resolución que tenga la imagen.

En la operación con el kernel plano 3x3 se pierde la primera y la última fila de la imagen, al igual que en las columnas, por lo tanto al realizar la convolución sería dos filas y dos columnas menos.

Opencv incorpora en sus librerías esta operación con el siguiente comando:

cv2.medianBlur(image,size(kernel)) [34]

En este proyecto se utiliza un kernel de 3x3, es decir size(kenel)=3, cada canal de la imagen se procesa independientemente.

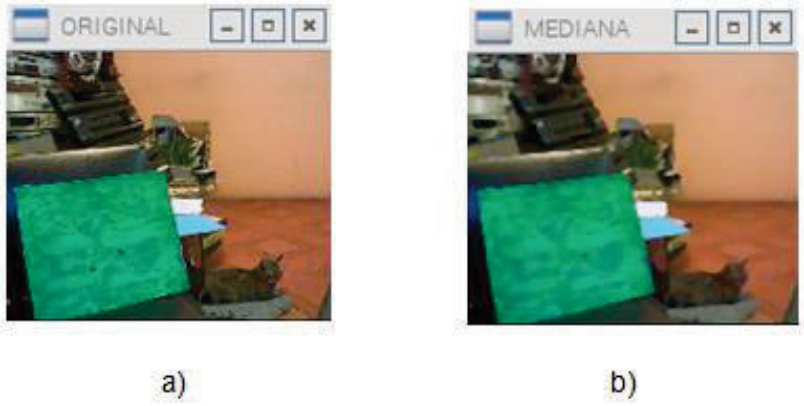


Figura 3.3. a) Imagen original b) Imagen aplicando el filtro de media

3.1.2.2 Filtro Gaussiano

El filtro Gaussiano es de tipo espacial lineal muy similar al filtro de media, pero el kernel ya no es de configuración plana, sino del tipo gaussiano que corresponde a la siguiente ecuación:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-u)^2/(2\sigma^2)} \tag{3.6}$$

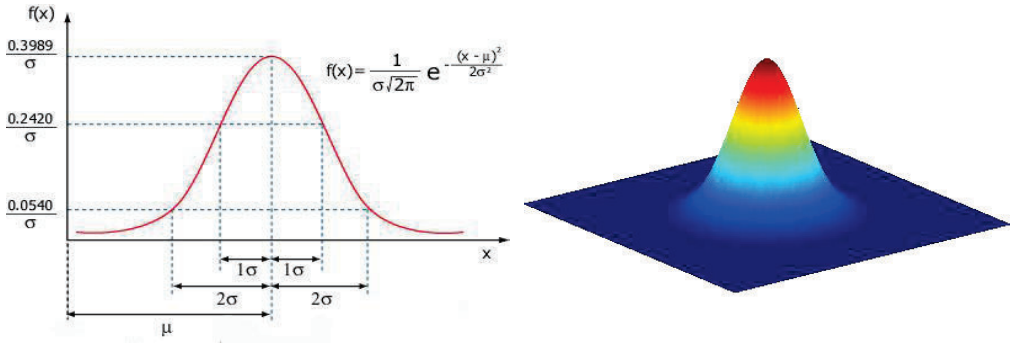


Figura 3.4. Función Gaussiana

Como se puede observar en la Figura 3.4 este tipo de filtro no es plano, se tiene valores más altos en la medida en que se va acercando al centro del pixel y van bajando proporcionalmente a los extremos, un ejemplo de kernel 5x5 gaussiano sería el siguiente si sigma ($\sigma = 1$):

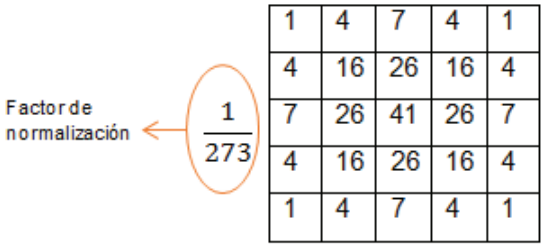


Figura 3.5. Kernel Gaussiano 5x5

El factor de normalización sirve para que los pixeles no vayan a saturarse, el valor se toma de la suma de los elementos de la matriz, como ejemplo el valor obtenido es 273.

Ya obtenido el kernel gaussiano se utiliza la ecuación (3.5) correspondiente a la de convolución, y se procede con el producto punto a punto de acuerdo con la Figura 3.2 para luego realizar la suma de la ecuación ya antes mencionada.

El filtro gaussiano es de tipo pasa bajos que permite eliminar ruido gaussiano que el filtro de media no lo realiza efectivamente.

Opencv incorpora en sus librerías esta operación con el siguiente comando:

cv2.GaussianBlur(entrada,sizekernel,Sigmax,Sigmay) [34]

Donde se debe de especificar los siguientes aspectos: la entrada a realizar el filtro gaussiano, el tamaño al cual se refiere el ancho y alto de la matriz del kernel que debe ser impar y positivo tomando el tamaño de (5,5) en este proyecto, las desviaciones estándar tanto en x (Sigmax) como en y (Sigmay), si se especifica solamente Sigmax, Sigmay tomará el mismo valor de Sigmax, si ambos se dan como 0 las sigmas se calculan automáticamente a partir del tamaño del kernel con la siguiente ecuación: [35]

$$sigma_{x\,e\,y} = 0{,}3 \cdot \big((sizekernel) \cdot 0{,}5 - 1\big) + 0{,}8 \qquad (3.7)$$

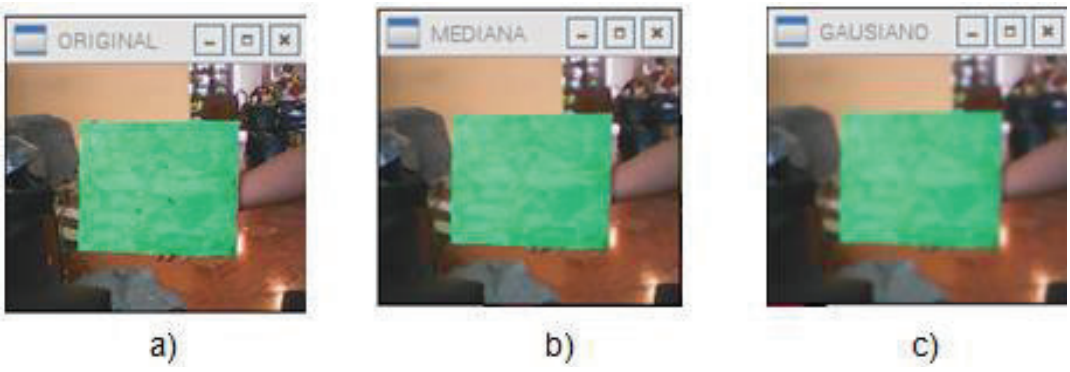


Figura 3.6. a) Imagen original b) Imagen aplicando el filtro de media c) filtro Gausiano

3.1.2.3 Conversión de BGR a HSV

Para poder identificar el color de manera más efectiva lo recomendable es pasar de espacio original (BGR) a espacio HSV, el cual proporciona valores de tono que va de 0° a 360°, saturación de 0% a 100% y brillo de 0% a 100%.

En este proyecto se utiliza el espacio HSV ya que se tiene un ambiente no controlado en el cual tanto la luminosidad como la saturación no son constantes.

Para realizar la conversión del espacio BGR a HSV se utilizan las ecuaciones (1.11) hasta (1.14)

Opencv incorpora en sus librerías dicha conversión con el siguiente comando:

cv2.cvtColor(entrada, tipo de conversión) [34]

Existen varios transformadores de espacio pero los más relevantes son: de BGR a GRAY (cv2.cv.CV_BGR2GRAY) y del proyecto de BGR a HSV (cv2.cv.CV_BGR2HSV).

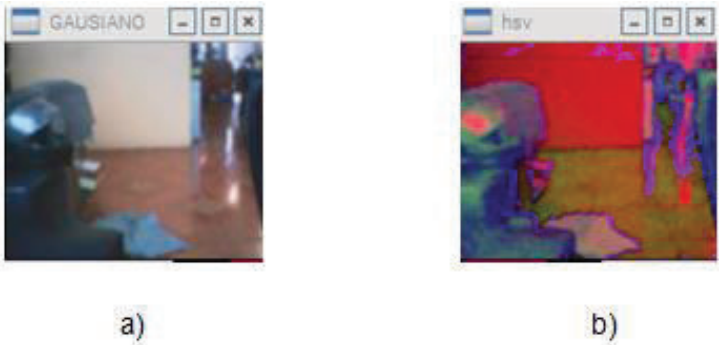


Figura 3.7. a) Imagen Gaussiana aplicada b) Conversión a HSV

3.1.3 SEGMENTACIÓN

El proceso de segmentación ayuda a reconocer el objeto por la textura, color, forma, detalles etc. Para cada caso existen diferentes técnicas.

En este proyecto se ha realizado una segmentación por color, la técnica utilizada de segmentación es la umbralización el cual realiza la conversión de la imagen en niveles de gris o de color en una imagen binaria; de tal manera que la región de interés los pixeles se pinten de blanco, dependiendo de la detección de color que se requiera realizar. El procesamiento necesario para realizar la umbralización es bajo para los computadores convencionales. [33]

3.1.3.1 Umbralización Fija

La umbralización fija consiste en colocar un rango que marque un umbral de separación con la imagen ya procesada. Se debe tener previamente los valores de la cantidad de tono, saturación y brillo del objeto a detectar, para luego realizar una umbralización dentro del espacio HSV, de esta manera obtenemos una imagen binaria con el objeto de detección. La umbralización cumple con la siguiente función: [33]

$$P(x,y) = \begin{cases} 1 & si \quad Umin < I(x,y) \leq Umax \\ 0 & si \quad otro\ caso \end{cases} \tag{3.8}$$

Donde $P(x,y)$ corresponde a la imagen binaria, $I(x,y)$ es la imagen original (HSV) y U el umbral dado por el usuario.

En este proyecto se realiza la detección de tres colores: verde, azul, y fucsia obteniendo una máscara previamente con el rango de tono, saturación, brillo dependiendo del color a detectar. Se realizaron pruebas para obtener el rango de valores del umbral, el cual se coloca en la siguiente tabla con las marcas ya determinadas:

Tabla 3.2. Rango de valores para la umbralización

Color a detectar	H	S	V
Verde	Hmin=55 Hmax=85	Smin=66 Smax=255	Vmin=79 Vmax=255
Azul	Hmin=86 Hmax=120	Smin=80 Smax=238	Vmin=80 Vmax=255
Fucsia	Hmin=160 Hmax=180	Smin=105 Smax=255	Vmin=113 Vmax=255

Los valores en la tabla son obtenidos en un ambiente no controlado con luminosidad media. Gracias a un comando de opencv **cv2.getTrackbarPos( )** el cual permite configurar los valores de tono, saturación y brillo cuando el programa se esté ejecutando, al tener la gran ventaja de poder observar y al mismo tiempo poder configurar hasta que la detección haya finalizado.

Se debe tomar en cuenta que los valores de umbralización son dados en bits de 0 a 255 los cuales deben pasar por una transformación previa, ya que H viene en grados, V y S vienen en %. Generalmente ya existen tablas o programas proporcionan los valores de color en HSV de objetos ya predeterminados pero en sistemas reales se debe de ajustar al medio no controlado.

Figura 3.8. a) Imagen HSV color verde b) Mascara de umbralización color verde

3.1.4 PROCESAMIENTO BASADO EN SUPERFICIE

La imagen segmentada contiene errores debido a que el color de la marca no es uniforme, o simplemente la luminosidad no es la misma en todos los puntos; para ello se utilizan filtros morfológicos que ayudan a minimizar los errores obtenidos por el proceso de umbralización.

3.1.4.1 Morfología

Se refiere a estudio de figuras, formas y estructuras cuyo fundamento matemático se basa en la teoría de conjuntos el cual fue introducida por Georges Matheron posteriormente Jeam Serra la extendió al análisis de imágenes. [36]

Considerando que la morfología matemática es una técnica de procesamiento no lineal la cual se caracteriza por realizar la geometría de objetos a detectar.

"El objetivo de las transformaciones morfológicas es la extracción de estructuras geométricas de los conjuntos sobre los que se opera, mediante la utilización de otro conjunto de forma conocida, al que se denomina elemento estructurante." [36]

Con respecto al elemento estructurante se debe elegir el tamaño y forma acorde al objeto que se va a aplicar y de acuerdo a lo que se desea extraer. El elemento estructurante se desplaza sobre la imagen acorde al tamaño de los pixeles.

Las operaciones morfológicas tienen diferentes aplicaciones como: filtrado de ruido, ampliación, reducción, afilado de objetos, etc.

Las operaciones morfológicas binarias más elementales son: erosión y la dilatación. Los filtros morfológicos son: apertura y cierre.

En este proyecto se aplicó la "erosión" como operación morfológica y el "de cierre" como filtro morfológico, ya que debido a la aparición de ruido en la imagen binarizada, estos permitieron eliminarlo.

3.1.4.1.1 Erosión

Al momento que se transforma la imagen original a una imagen binaria en el proceso de segmentación, se puede trabajar mediante conjuntos.

Se tiene la imagen que corresponde al conjunto (A) la cual va a hacer erosionado y por otro lado el elemento estructurante o kernel (B) el mismo que se define como el conjunto de todos los puntos z pertenecientes a A, de tal forma que el elemento estructurante B se traslada a ese punto.

$$\varepsilon_B(A) = A\Theta\mathrm{B}\{z|(B)_z \subseteq A\} \tag{3.9}$$

El resultado de la erosión es comprobar si el conjunto B está completamente incluido en el conjunto A, si esto no ocurre el resultado nos proporciona un conjunto vacío. [36]

Se puede reformular la ecuación (3.9) como una intersección de conjuntos trasladados donde las mismas se definen por el elemento estructurante con la siguiente ecuación:

$$A\Theta\mathrm{B} = \cap_{b\in B}\, A_{-b} \tag{3.10}$$

Donde $A\Theta\mathrm{B}$ es la erosión del conjunto A (imagen) con el B (kernel) el cual es la intersecciones de todas las traslaciones de A por los puntos $-b$ donde $b \in B$. [36]

OpenCv cuenta con un comando que realiza la intersección de la imagen con el elemento estructurante en todos los puntos, el cual es el siguiente:

cv2.erode(entradabinaria, kernel, iterations=3) [34]

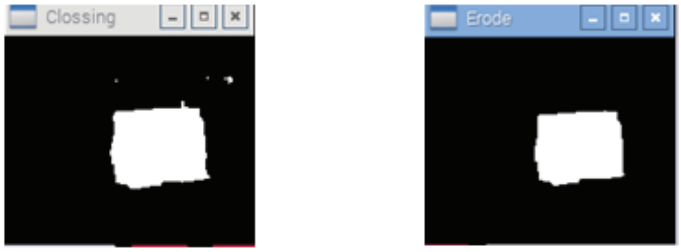
Este comando permite realizar la erosión en los límites siempre tratando de mantener el primer plano en blanco. El kernel se desliza sobre la imagen, el pixel de la imagen se considera 1 solo si todos los elementos del kernel son 1 de lo contrario se erosiona, es decir; los pixeles cerca del límite son descartados dependiendo del tamaño del kernel, el resultado será una imagen con el grosor y tamaño disminuido. [35]

a) b)

Figura 3.9 a) Imagen original b) Imagen aplicando la erosión [35]

Se utiliza esta operación ya que existen ruidos externos que deben ser erosionados.

En este proyecto se usa una matriz kernel de 3x3 con un total de 3 interacciones y se aplica la erosión después de la realización del filtro de cierre que se explicará a continuación:



a) b)

Figura 3.10. a) imagen aplicada la erosión b) imagen erosión

3.1.4.1.2 Dilatación

"Es el conjunto de puntos origen del elemento estructurante B tales que el elemento estructurante complementado contiene algún elemento del conjunto A, cuando el elemento se desplaza por el espacio contienen a ambos conjuntos" [36]

$$\delta_B(A) = A \oplus B = \{z | (\hat{B})_z \cap A \neq \emptyset\} \tag{3.11}$$

Se puede reformular la ecuación (3.11) como una unión de conjuntos trasladados donde las mismas se definen por el elemento estructurante con la siguiente ecuación:

$$A \oplus \mathrm{B} = \cup_{b \in B} A_b \tag{3.12}$$

La dilatación produce una extensión de la imagen original ya que es la unión entre el elemento estructurante y la imagen.

a) b)

Figura 3.11. a) Imagen original b) Imagen aplicando la dilatación [35]

Se menciona la dilatación ya que el filtro de cerrado utilizado en este proyecto es una mezcla entre erosión y dilatación.

3.1.4.1.3 Filtro de cierre

El filtro de cierre de la imagen A por el elemento estructurante (kernel) B se define como la dilatación entre A y B, seguida de la erosión del mismo elemento estructurante.

$$\varphi_B(A) = A \bullet \mathrm{B} = (A \oplus \mathrm{B}) \ominus \mathrm{B} = \varepsilon_B\big(\delta_B(A)\big) \tag{3.13}$$

"Si A no cambia con el cierre por B se dice que A es cerrada respecto a B" [36]

La aplicación del filtro de cierre es una combinación entre dilatación y erosión. Al momento que se rellenan los huecos interiores de la imagen aplicando la dilatación esta aumenta el tamaño del objeto, por ende se realiza la erosión, el cual devuelve a su tamaño inicial. El filtro morfológico de cerrado permite rellenar imágenes sin que estas se vean afectados por el tamaño. [36]

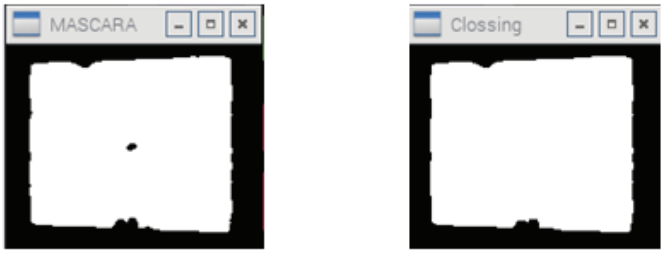
OpenCv proporciona un comando para la realización del filtro de cerrado el cual es el siguiente:

cv2.morphologyEx(entradabinaria, cv2.MORPH_CLOSE, kernel) [34]

Es usualmente utilizado para eliminar pequeños agujeros en el interior del objeto, al igual que la erosión se utiliza un kernel que se desliza en la imagen realizando la dilatación para posteriormente realizar la erosión.

Figura 3.12. Imagen aplicando el filtro de cierre

En el proyecto se realiza primero el filtro de cierre y posteriormente la erosión.



a) b)

Figura 3.13. a) Imagen aplicada el filtro cerrado b) Imagen con el filtro de cierre

3.1.5 RECONOCIMIENTO

En el procesamiento de imágenes se detectó el color, en este proyecto se tiene tres marcas con figuras conocidas que son: cuadrado, triángulo y círculo, por lo que para el reconocimiento del tipo de marca se usa el reconocimiento de contornos y de esta forma se conoce el número de vértices en el caso del cuadrado y triángulo. Para el círculo se usa otro proceso diferente que se explicará más adelante.

3.1.5.1 Detección bordes

Para realizar la detección de bordes se necesita tener un filtro de obtención de contornos, existen tres tipos de filtros Sobel, Roberts y Prewitt.

La derivada direccional de una función permite determinar los cambios bruscos de intensidad.

El operador gradiente se define como una función de la siguiente ecuación:

$$\nabla\big(I(x,y)\big) = \frac{dI}{dx}\overrightarrow{u_x} + \frac{dI}{dy}\overrightarrow{u_y} \tag{3.14}$$

Donde $\frac{dI}{dx}$ y $\frac{dI}{dy}$ se definen como filtrados de convolución Gx y Gy respectivamente el cual se expresa por las siguientes ecuaciones:

$$Gx = \frac{dI}{dx} = I(x,y) * h_1(x,y) \tag{3.15}$$

$$Gy = \frac{dI}{dy} = I(x,y) * h_2(x,y) \tag{3.16}$$

Donde $I(x,y)$ es la imagen en pixeles que se convolucionan con las matrices h1 y h2 llamados “ventana de Sobel”. [33]

$$h1 = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \; h2 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{3.17}$$

Los cambios de claro a oscuro se marcan con un valor positivo y los cambios de oscuro a claro se marcan con un valor negativo, mediante $h1$ y $h2$ se calcula el gradiente en las direcciones vertical y horizontal respectivamente. [33]

Se mencionaran el filtrado de Robert y Prewitt para conocimiento general la única diferencia son el cambio de matrices $h1$ y $h2$ en los dos casos.

$$Prewitt: \quad h1 = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \; h2 = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ .1 & 0 & 1 \end{pmatrix} \tag{3.18}$$

$$Robert: \; h1 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad h2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{3.19}$$

La detección de bordes de Canny formulada por John F. Canny en 1986, el cual tiene las siguientes etapas:

3.1.5.1.1 Eliminación de Ruido

El principal problema de la detección de bordes de Canny es el ruido por lo que puede detectar falsos bordes, para ello se recomienda realizar un filtro gaussiano ya que es el más óptimo, hay que tener cuidado de que σ no sea muy grande ya que se puede perder todos los contornos.

En este proyecto no fue necesario realizar el filtro gaussiano ya que se tiene una imagen binarizada aplicada los filtros adecuados para una correcta detección y sin ruidos.

3.1.5.1.2 Encontrar la intensidad del gradiente de la imagen

Una vez que se tiene la imagen filtrada y lista para aplicar la detección de bordes, se debe encontrar la intensidad que tiene un borde a través del gradiente de la imagen. La imagen binarizada se filtra con el kernel de Sobel 3x3 según matriz (3.17) en dirección vertical y horizontal para obtener la derivada en dirección vertical Gy y en dirección horizontal Gx. A partir de estas dos imágenes se puede

encontrar el gradiente del borde y la dirección para cada pixel de la siguiente forma: [35]

$$Gradiente\ borde\ (G) = \sqrt{G_x^2 + G_y^2} \tag{3.20}$$

La dirección es siempre perpendicular a los bordes.

Si Gx es igual a cero, la dirección del borde tiene que ser igual a 90° o 0°. Si Gy tiene un valor de cero, la dirección del borde será igual a 0°, de lo contrario, la dirección del borde será igual a 90°. La fórmula para encontrar la dirección del borde es la siguiente: [37]

$$Angulo\ (\theta(x,y)) = tan^{-1}(\frac{G_y}{G_x}) \tag{3.21}$$

3.1.5.1.3 Supresión no máxima

Después de haber obtenido la magnitud y dirección del gradiente, se realiza un análisis de la imagen para eliminar los pixeles que no constituyen el borde.

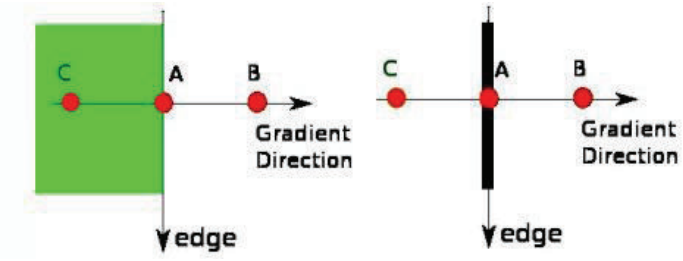


Figura 3.14. Supresión máxima [35]

En la Figura 3.14 el punto A está en el borde en sentido vertical el cual la dirección del gradiente es perpendicular al borde. Los puntos B y C están en direcciones del gradiente, de esta manera el punto A está marcada con los puntos B y C para ver si forma un máximo local. Si forma un máximo local se pasa a la siguiente etapa de lo contrario se suprime (coloca un 0). Esto dará una delgada línea en el borde de salida.

3.1.5.1.4 Umbral de histéresis

Esta es la última etapa para la detección de Canny en la cual se necesita dar dos valores de umbral un valor mínimo (minVal) y un valor máximo (maxVal).

Cualquier borde con gradiente de intensidad por encima del maxVal representa que si existe un borde, por debajo del minVal representa que no existe un borde por ende lo desecha y los que están entre el rango entre minVal y maxVal puede tener un borde o no, depende de su conectividad con referencia al maxVal. [35]

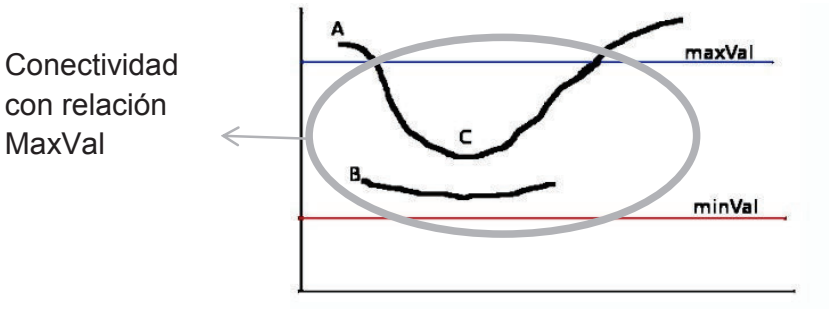


Figura 3.15. Rango valores mínimo (MinVal) y máximo (MaxVal) del algoritmo de Canny.

Como se puede observar en la Figura 3.15 se tiene tres bordes A, B y C el cual se determinará si son bordes o falsos bordes. Se observa que A se encuentra por encima de maxVal por ende si es un borde, el B se encuentra dentro del límite entre maxVal y minVal el cual se determina si es borde falso o no por su conectividad, como no llega la conectividad hacia el maxVal no es un borde, por el contrario el C también se encuentra dentro de los límites pero este si tiene conectividad hacia el maxVal por ende si es un borde.

OpenCv ha incorporado la función Canny en un solo comando que es el siguiente:

cv2.Canny(entrada,minVal,maxVal,sizekernel) [34]

El primer argumento es la entrada al cual se va a realizar la función canny, el segundo y tercer argumento son los límites del umbral de histéresis, el cuarto argumento es el tamaño del kernel de Sobel, si ese argumento no se encuentra escrito por defecto es 3x3. Se usa la ecuación (3.20) referente a la obtención del gradiente.

Una vez que se realiza la detección de bordes se debe encontrar el contorno de la figura ya que tiene el mismo color e intensidad. Para una mayor facilidad se considera que la figura debe ser de color blanco y de fondo negro (imagen binaria).

OpenCv tiene una función para encontrar los contornos el cual es la siguiente:

cv2.findContours (entrada,modo,método) [34]

El primer argumento es la imagen binarizada, el segundo el modo de recuperación de contorno y el tercero el método de aproximación del mismo. Los tipos de modo y método que se pueden usar se muestran en la siguiente tabla: [34]

Tabla 3.3. Modo y método de función opencv findContours

Modo		**Método**	
CV_RETR_EXTERNAL	Recupera solo el contorno exterior extremo	**CV_CHAIN_APPROX_NONE**	Almacena todos los puntos del contorno
CV_RETR_LIST	Recupera todos los contornos sin establecer jerarquía	CV_CHAIN_APPROX_SIMPLE	Comprime los segmentos horizontales, verticales y diagonales y deja solamente sus puntos finales.
CV_RETR_CCOMP	Recupera todos los contornos y las organiza en dos jerarquías	CV_CHAIN_APPROX_TC89_L1, CV_CHAIN_APPROX_TC89_KCOS	se aplica uno de los algoritmo favoritos de aproximación cadena Teh-Chin
CV_RETR_TREE	Recupera todos los contornos y reconstruye una jerarquía completa de los contornos anidados.	Offset	Opcional compensado por la que se desplaza cada punto del contorno

En este proyecto se utiliza el modo CV_RETR_LIST ya que se necesita recuperar todos los contornos sin considerar la jerarquía y el método CV_CHAIN_APPROX_SIMPLE porque no se requiere todos los puntos sino nada más de los finales, para que no se ocupe toda la memoria.

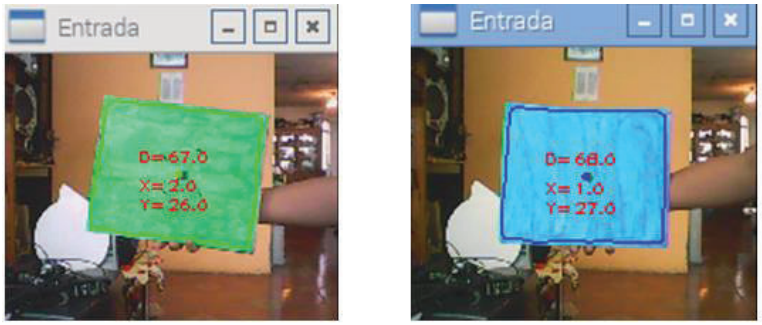


Figura 3.16. a) Contorno figura verde b) Contorno figura azul

La Figura 3.16 se muestra el resultado de la detección de contornos para una figura de color verde y una azul, se observa que el comando funciona apropiadamente de acuerdo a los parámetros utilizados en el mismo.

3.1.5.2 Detección de figuras

Una vez realizado la recuperación de contornos con los puntos de interés se realiza la detección de la figura sea cuadrado, triángulo y círculo.

Un método para detectar el triángulo y cuadrado es por medio de los vértices ya que los mismos cuentan 3 y 4 vértices respectivamente.

Para la detección de círculos se utiliza otro método llamado la "transformada de hough" aplicada a círculos.

3.1.5.2.1 Identificación de vértices

Para las detecciones de los vértices se utiliza el siguiente comando de OpenCV:

cv2.approxPolyDP(figura,0.05·cv2.arcLength(figura,True),True) [34]

Este comando usa el algoritmo de Douglas-Peuker el cual reduce el número de puntos aproximados en una curva a una la distancia $\varepsilon > 0$, en otras palabras busca el punto más alejado del segmento constituido por los puntos inicial y final de la curva, si el punto se encuentra más cerca del segmento a comparación de la distancia ε, pues este punto no se simplifica caso contrario si lo hará.

El primer argumento de este comando es la imagen binarizada, el segundo es el ε que es la distancia máxima del contorno aproximado y es un parámetro de precisión, se necesita conocer bien este parámetro para una detección correcta del vértice.

En este proyecto se toma el ε con **cv2.arcLength** que calcula el perímetro del lado de la curva, el mismo se multiplica por 0.05, valor obtenido de pruebas realizadas, con el cual se puede detectar los vértices correctamente.

El tamaño del vector de salida del comando es el número de vértices que se tiene en la imagen.

Si el tamaño del vector es 4 se reconoce como cuadrado y si es 3 es triángulo.

3.1.5.2.2 Transformada de Hough para círculos

La transformada de Hough es un algoritmo de votación resultante del conjunto de puntos que conforman una recta o círculo cuyo espacio de parámetros es de 2 y 3 dimensiones respectivamente.

En el proyecto se utiliza la transformada de Hough para detectar círculos, dada por la ecuación de la circunferencia que es la siguiente:

$$(x-a)^2 + (y-b)^2 = r^2 \tag{3.22}$$

Donde (a, b) son coordenadas del punto centro del círculo y r el radio del mismo a continuación (x, y) son puntos arbitrarios que pertenecen al contorno, el cual se transforma al espacio de un cono circular recto de parámetros de 3 dimensiones (a, b, r), que son las siguientes:

$$a = x - r \cdot cos\theta \tag{3.23}$$

$$b = y - r \cdot cos\theta \tag{3.24}$$

Si todos los puntos de la imagen se encuentran en un círculo entonces los conos se intersecan en un único punto (a, b, r) correspondiente a los parámetros del círculo. [38]

La matriz de acumulación de tres dimensiones utiliza la información de la dirección del borde el cual sirve para limitar la votación de una sección del cono mostrado en la Figura 3.17. En situación normal, el centro del círculo debe estar en una línea perpendicular a la dirección del borde, por lo tanto sólo se tiene que mover a lo largo de la normal de cada punto de borde para encontrar las posibles ubicaciones de los centros. La distancia entre cada punto del borde y el centro estimado es un candidato para el radio del círculo correspondiente. [38]

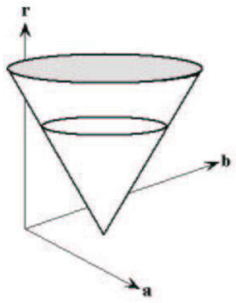


Figura 3.17. Espacio de parámetros Transformada Hough [39]

La mejor estrategia para la extracción de círculo es la siguiente:

- Detección de bordes mediante técnica de Canny.
- "Establecimiento de un dominio de parámetros cuyas dimensiones sean el propio espacio de búsqueda y una cuantización suficientemente precisa: la de los píxeles de la imagen original". [40]
- Se analiza toda la imagen de manera que cada pixel de borde da lugar a un radio (r) centrado. Las celdas que pertenecen al círculo tienen un voto.
- Todos los pixeles que pertenecen al círculo, en el espacio de parámetros se cortan en la misma celda de manera que el centro de cada círculo se determina por la celda más votada. [40]

OpenCv cuenta con un comando para la realización de la transformada de Hough aplicada para la extracción de círculos el cual es la siguiente:

cv2.HoughCircles(image, method, dp, minDist[, circles[, param1[, param2[, minRadius[, maxRadius]]]]]) [34]

Dónde:

image: Imagen de entrada binarizada

method: método de detección usada, se ocupa CV_HOUGH_GRADIENT

dp: Razón inversa de la resolución del acumulador a la resolución de la imagen. Por ejemplo, si dp = 1, el acumulador tiene la misma resolución que la imagen de entrada. Si dp = 2, el acumulador tiene la mitad del tamaño anchura y altura.

minDist: La distancia mínima entre los centros de los círculos detectados.

param1: En caso de CV_HOUGH_GRADIENT, es el umbral más alto de los dos pasó a la Canny () detector de borde (el inferior es dos veces más pequeño).

param2: En caso de CV_HOUGH_GRADIENT, es el umbral acumulador para los centros de círculo en la etapa de detección.

minRadius: radio mínimo.

maxRadius: radio máximo.

La salida de este comando puede ser un conjunto vacío o lleno, de esta manera se puede conocer si existen círculos o no.

3.2 MEDICIONES EN UNA IMAGEN

3.2.1 MOMENTOS GEOMÉTRICOS

Los momentos geométricos son los descriptores más simples dentro de las funciones de los momentos, los cuales proporcionan información con muchas aplicaciones en detección y estimación de la pose de un objeto en una imagen. Son también llamados momentos cartesianos o momentos regulares. [41].

Los momentos de orden $(p+q)$ de un área *A* dentro de una imagen continua $f(x,y)$ [42], son definidos como:

$$M_{pq} = \iint_{A}^{A} x^p y^q f(x,y) d_x d_y \tag{3.25}$$

Donde *p, q*: pertenecen al conjunto infinito de los números naturales.

Para una imagen digital de área *A* la ecuación (3.5) quedaría expresada de la siguiente manera [42].

$$M_{pq} = \sum_{(x,y)\in A} x^p y^q f(x,y) \tag{3.26}$$

3.2.1.1.1 Momentos Centrales

El momento simple de orden cero $M00$, representa el área en imágenes binarias y la superficie en imágenes en escala de grises. Es otras palabras es la suma de los valores de todos los pixeles. [43]

$$M_{00} = A = \sum_{(x,y)\in A} x^0 y^0 f(x,y) = \sum_{(x,y)\in A} f(x,y) \tag{3.27}$$

En la ecuación (3.27) se puede apreciar que se realiza la sumatoria de los valores de todos los pixeles que forman el objeto obteniendo de esa manera el área detectada.

En cambio para encontrar el centro de masa (centroide) de una imagen se utiliza los momentos simples de primer orden $M01$ y $M10$, como puede observarse en la ecuación (3.28) [43]

$$\bar{x} = \frac{M_{10}}{M_{00}} \qquad y \qquad \bar{x} = \frac{M_{01}}{M_{00}} \tag{3.28}$$

Para el efecto OpenCV pone a disposición el comando *moments*, el cual encuentra todos los momentos simples que pertenecen al objeto dentro de la imagen, teniendo como parámetro del comando, la imagen binarizada.

cv2.moments (imagen binarizada)

Como resultado obtenemos un vector con todos los momentos del objeto en la imagen.

3.2.2 MEDICIÓN DE PROFUNDIDAD Y DISTANCIAS

La medición de distancias ha sido realizada generalmente por medio de cámaras estéreo, sin embargo en este proyecto se utilizará una sola cámara para realizar tales mediciones.

3.2.2.1 Modelo de Cámara Pinhole o Estenopeica

La cámara Pinhole también conocida como cámara obscura es un dispositivo óptico con forma de caja o cámara. Posee un pequeño agujero en el cual por la propagación de la luz que lo atraviesa, se crea una imagen del espacio externo en el lado opuesto de la caja, como puede apreciarse en la Figura 3.18**.** [44]

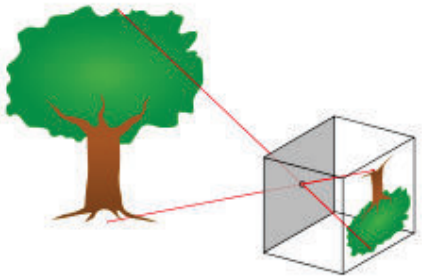


Figura 3.18. Representación del Modelo de la cámara Pinhole.

A continuación se presenta el modelo de la cámara pinhole en la Figura 3.19.

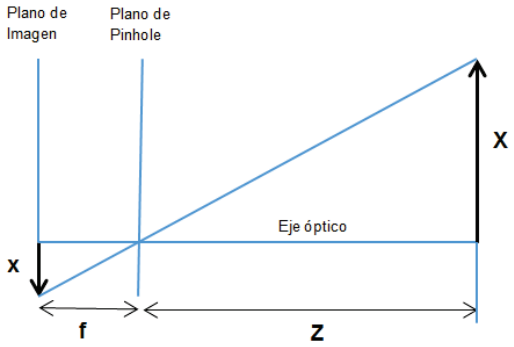


Figura 3.19. Modelo de cámara Estenopeica o Pinhole [45]

Podemos obviar el signo utilizando un modelo equivalente en donde todos los rayos parten del centro de proyección y la imagen no está invertida. [45]

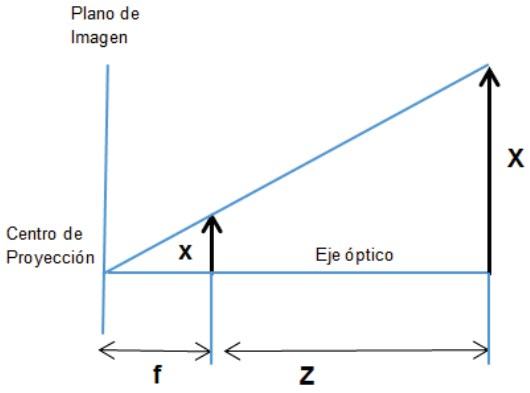


Figura 3.20. Modelo de cámara Pinhole sin inversión de imagen [45]

Aplicando geometría a los triángulos de la Figura 3.19 se puede determinar que existe semejanza entre ellos, obteniéndose la ecuación (3.29) que representa el modelo de la cámara Pinhole o Estenopeica, relacionando el tamaño del objeto visualizado con la distancia del mismo hacia la cámara.

$$\frac{f}{x} \approx \frac{Z}{X} \tag{3.29}$$

Dónde:

f: es la distancia focal

Z: distancia desde la cámara hacia el objeto

X: Tamaño real de la imagen

x: tamaño de la imagen proyectada

Como puede observarse la distancia focal de la cámara (f) es constante, no depende de la distancia a la que se encuentre la imagen. Tomando esto en cuenta se procede a colocar la imagen con tamaño real conocido (X) frente a la cámara a una distancia conocida (Z) y a través del cálculo del área con el momento M_{00} se calcula la altura del objeto (x). Con estos datos se determinó el valor de la distancia focal (f).

$$f = \frac{Z \cdot x}{X} \tag{3.30}$$

La distancia focal calculada para el cuadrado es $f = 2395$ y para el círculo es $f = 2197$.

Conociendo la distancia focal, es posible calcular para cada instante la distancia a la que se encuentra el objeto:

$$profundidad = \frac{f \cdot X}{x} \tag{3.31}$$

Este procedimiento fue realizado para las figura cuadrado y circulo.

Para el triángulo se realiza un procedimiento distinto ya que se tiene errores en la toma de datos. Se procede a diseñar una ecuación que cumpla con la distancia de acuerdo al área, encontrándose la siguiente ecuación:

$$profundidad = 100 \cdot 56.58 \cdot area^{-0.32} \tag{3.32}$$

Para determinar las distancias en los ejes X y Y se realizó ecuaciones dependiendo de la distancia a la que se encuentre el objeto, obteniéndose las siguientes:

$$distancia_x = 0.8862 \cdot profundidad - 0.017 \tag{3.33}$$

$$distancia_y = \frac{38.659 \cdot profundidad + 7.6912}{100} \tag{3.34}$$

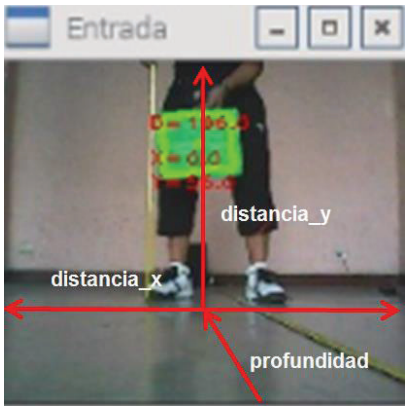


Figura 3.21. Representación de distancias en la captura de la cámara

Una vez obtenido la $distancia_y$ total dependiendo de la profundidad a la que se encuentra el objeto, se debe efectuar un par de operaciones que relacionen la resolución en altura (height) y la centroide del objeto detectado.

$$distancia = heigth - y \tag{3.35}$$

$$distanciay = \frac{distancia \cdot distancia_y}{height} \tag{3.36}$$

Donde $heigth = 144$ mencionado con anterioridad, y y es obtenida de la ecuación (3.38).

En la ecuación (3.36) se suma 11cm ya que la posición de la cámara referente al suelo se encuentra a esa distancia.

La $distancia_x$ ayuda a calcular el ángulo de desviación del robot con referencia al objeto, el mismo que se explica a continuación.

3.3 CONTROLADOR DE MOTORES

Debido a que el robot es de tracción diferencial se controlan los motores usando señales PWM.

El controlador utilizado es un (PID) similar a los controladores de los robots seguidores de línea ya que no cuentan con encoder de realimentación de la velocidad y solo se utiliza realimentación de posición.

El único parámetro de control es la posición, cuyo valor es el cálculo de la centroide del área de la imagen detectada gracias a los momentos cuyas ecuaciones son las siguientes:

$$x = \frac{momentos['m10']}{momentos\ ['m00']} \tag{3.37}$$

$$y = \frac{momentos['m01']}{momentos\ ['m00']} \tag{3.38}$$

Una vez obtenido el centroide, el valor se transforma de acuerdo a la resolución que con la que se esté trabajando, para este proyecto se trabaja en el eje x con una resolución de width= 170, en el cual la mitad de la pantalla es de 85 como puede observarse en la Figura 3.22.

Figura 3.22. Imagen de muestra con resolución width=170

El error que ingresa al controlador PID es el ángulo de desviación que dependerá de la profundidad, la resolución y la distancia X obtenida en la ecuación (3.33). Para normalizar el error respecto al centro de la imagen se usa la ecuación (3.39). Para relacionar la distancia X con respecto a la resolución en ancho se usa la ecuación (3.40).

$$x_{cua} = x - 85 \tag{3.39}$$

$$valorx = \left(\frac{x_{cua} \cdot distancia_x}{width}\right) \tag{3.40}$$

$$\theta = \arctan\left(\frac{valorx}{profundidad}\right) \tag{3.41}$$

Donde x es obtenida de la ecuación (3.37), $distancia\_x$ de la ecuación (3.33), $profundidad$ de la ecuación (3.31) y $width$ es la resolución de la pantalla en ancho que es 170.

La realimentación del lazo de control es la posición obtenida por la cámara, el elemento final es el driver el cual comandará los motores para que el robot sea dirigido hacia el centro del objeto.

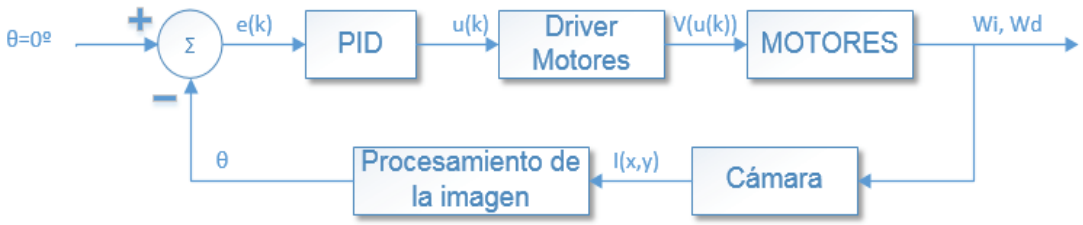


Figura 3.23 Diagrama de control de posición

El setpoint tiene que ser la mitad de la pantalla para realizar el seguimiento, por ende en este caso tiene que ser 0° de desviación, el valor de error que ingresa al PID es θ. Las constantes Kp, Kd y ki se obtuvieron de acuerdo a Ziegler y Nichols por el método de oscilación ya que no se tiene el modelo de la planta. Los valores obtenidos son los siguientes:

$$Kp = 1{,}08$$

$$Kd = 0{,}37$$

$$Ki = 0{,}66$$

De esa manera se obtiene el error y error anterior y se calcula el controlador.

$$u_k = Kp \cdot e_k + Kd * (e_k - e_{k-1}) + Ki \cdot (\sum (e_k - e_{k-1})) \quad (3.42)$$

Donde e_k es el error y e_{k-1} es el error anterior. El valor del controlador va en un rango de 0 a 100, pero el valor máximo que puede llegar la PWM es el 40% ya que para realizar una correcta detección el robot no debe ir muy rápido.

La salida de señal de las PWM son por los pines 11 y 18 correspondientes a los GPIO 17 y 24 respectivamente.

3.4 COMUNICACIÓN I2C

La comunicación I2C es un protocolo serial de comunicación usando pocos cables. El dispositivo maestro controla la comunicación I2C y se conecta todos los dispositivos esclavos en el mismo cable. Cada dispositivo esclavo tiene un único número llamado “dirección” y solo responde a mensajes que este recibe con la dirección que se especifica. Esto significa que no se puede usar dos dispositivos esclavo que tengan la misma dirección.

Las dos líneas I2C son llamadas SDA y SCL. Se necesita usar una resistencia en estas líneas, el valor de las resistencias no es importante pero se recomienda una resistencia que esté alrededor de 10KΩ.

I2C puede ser es conveniente en la raspberry pi ya que solamente tiene que convertir el nivel lógico del cable SDA para interactuar con el dispositivo de 5V.

Cualquier pin del GPIO puede ser usado como I2C, si el usuario requiere escribir todo el código, pero es un protocolo más complejo de implementar. Para usar las librerías y módulos ya preconstruidas del I2C, los pines 3 y 5 del GPIO son usados para la utilización de este protocolo.

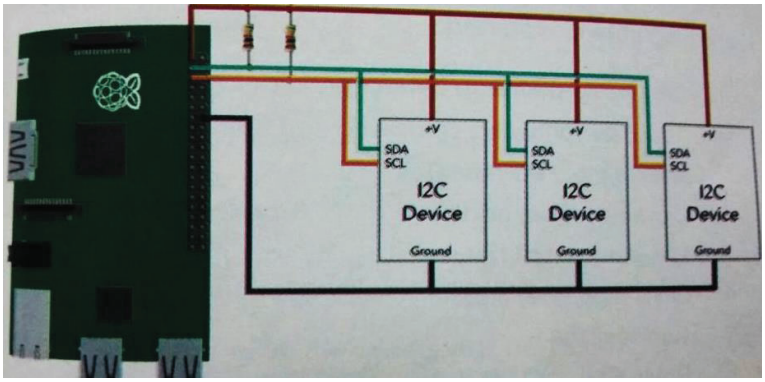


Figura 3.24. Conexión raspberry pi con dispositivos esclavos

En el proyecto no existen la resistencia conectadas físicamente ya que internamente se activó los pull-up a las salidas de los GPIO correspondiente a SDA y SCL.

Para enviar datos por I2C se habilita la librería SMBUS con el comando import.

El dispositivo esclavo, quien se va a comunicar con la raspberry pi, es un arduino nano el cual recoge el dato enviado por un vector transformado en string y lo muestra en un LCD, la dirección del arduino nano es la 0x07. El comando utilizado para enviar el dato es el siguiente:

bus.write_i2c_block_data(i2c_direccion,i2c_cmd,dato)

Donde se especifica la dirección del esclavo y el dato a ser enviado, como en este proyecto se debe enviar varios datos, se prefirió colocarle en un vector para luego transformarle a string y posteriormente a bytes.

3.5 DESARROLLO DEL SOFTWARE

A continuación se presentan los diagramas de flujo que representan la lógica de programación seguida para la realización de este proyecto.

3.5.1 FUNCIONES DE MOVIMIENTO

El siguiente diagrama muestra la función de giro del robot al momento de inicializar el programa, reconociendo el entorno y las figuras que se encuentren en él.

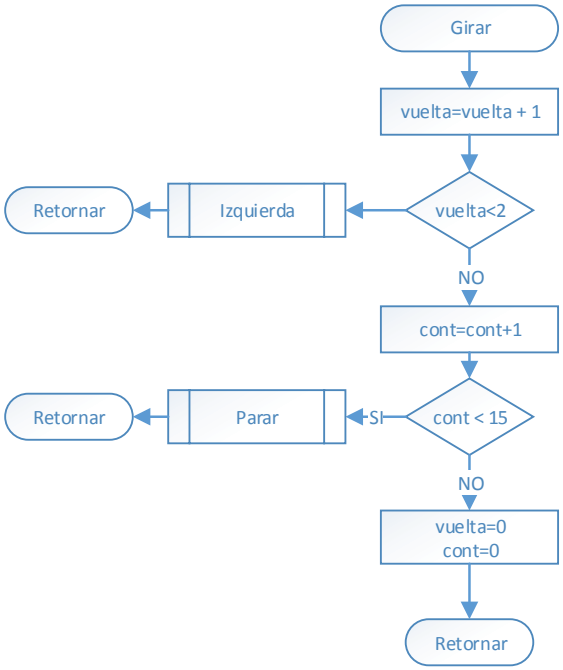


Figura 3.25. Función para el movimiento de giro temporizado

El siguiente diagrama muestra el envío de la señal (PWM) a los motores para realizar el giro izquierdo, por tracción diferencial.

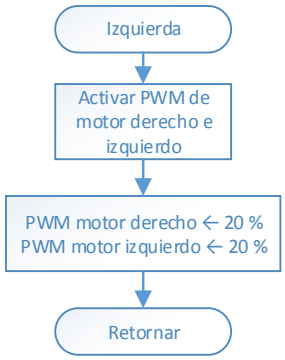


Figura 3.26. Giro izquierdo

El siguiente diagrama muestra el envío de la señal (PWM) a los motores para realizar el giro hacia atrás.

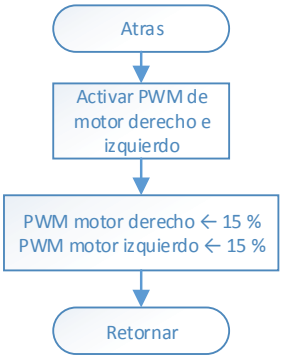


Figura 3.27. Función para el movimiento hacia atrás

El siguiente diagrama muestra el envío de la señal (PWM) a los motores con un valor de cero.

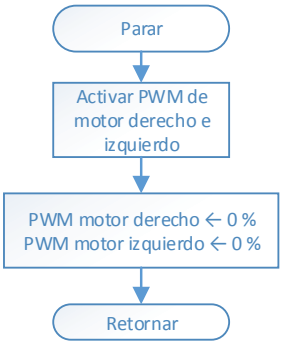


Figura 3.28. Función para parar el robot

El siguiente diagrama muestra el envío de la señal (PWM) a los motores para realizar el seguimiento del objeto.

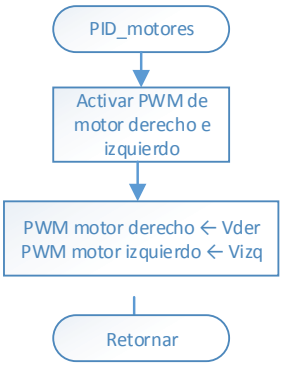


Figura 3.29. Función de envío de los valores calculados del controlador PID

3.5.2 FUNCIONES DE PROCESO

El siguiente diagrama muestra el cálculo de los valores de las señales PWM de acuerdo al ángulo de desviación del objeto.

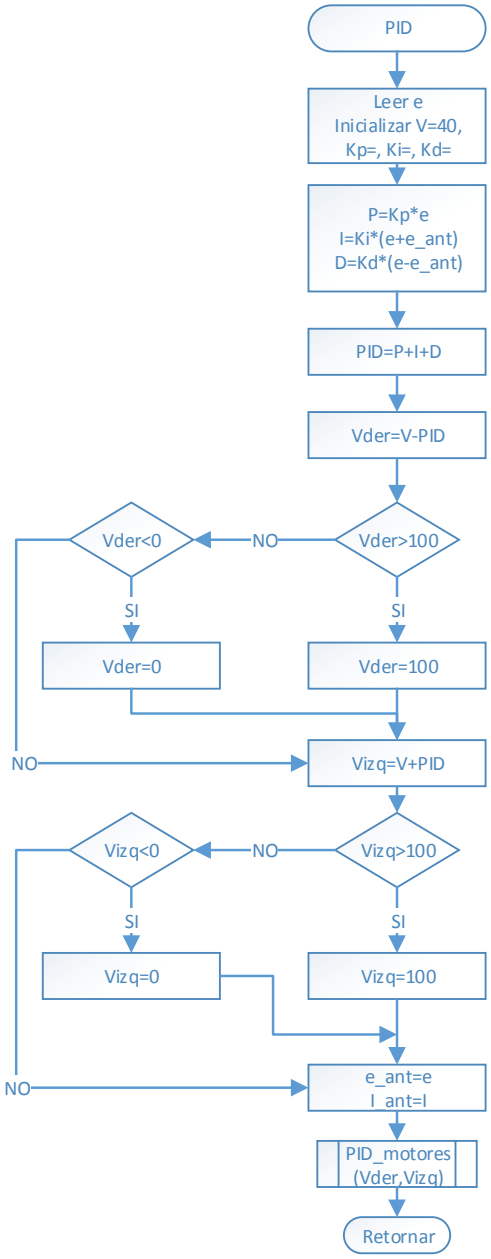


Figura 3.30. Función del controlador PID

El siguiente diagrama muestra los algoritmos de filtrado y segmentación.

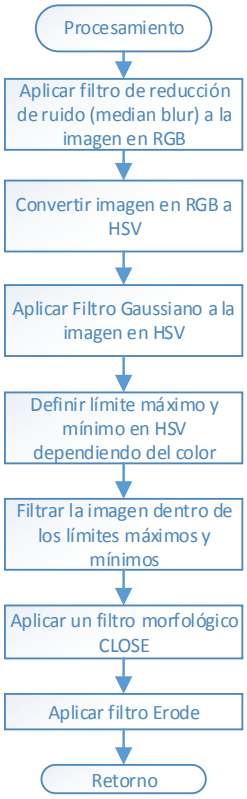


Figura 3.31. Algoritmo de filtrado y segmentación

El siguiente diagrama muestra el algoritmo de detección de contornos.

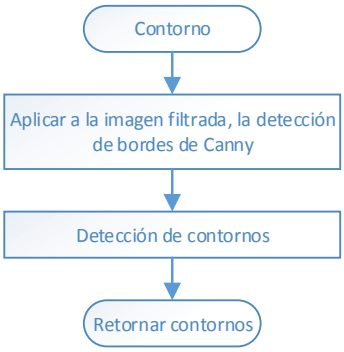


Figura 3.32. Función de detección de contornos

3.5.3 PROGRAMA PRINCIPAL

El programa principal llama a las funciones antes definidas en un orden específico. Las tareas que se realizan son: adquisición de imagen, filtrado, segmentación, identificación, reconocimiento, seguimiento, medición de distancias y transmisión de datos. Los objetos a detectar son de colores verde, azul y fucsia, con las formas cuadrado, triángulo y círculo.

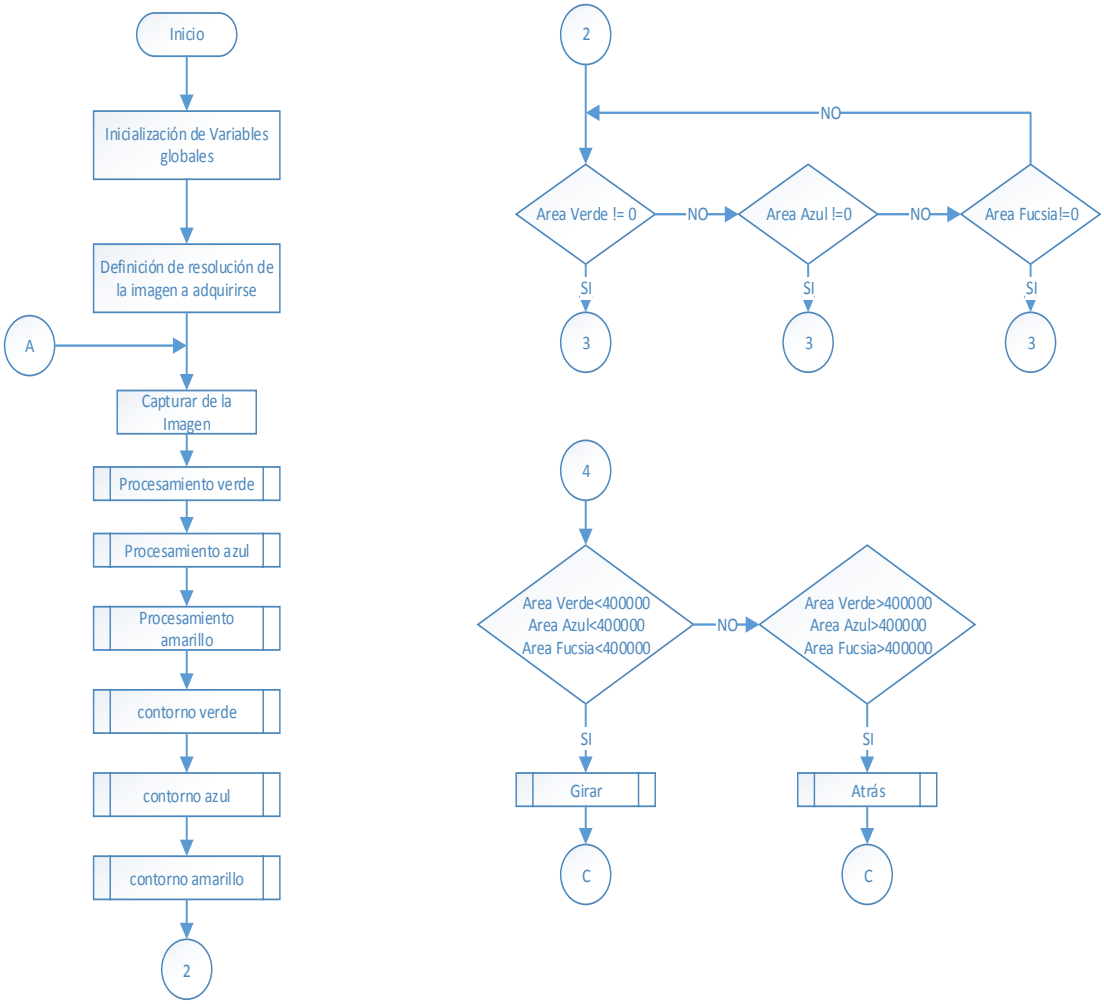


Figura 3.33. Programa principal, parte 1

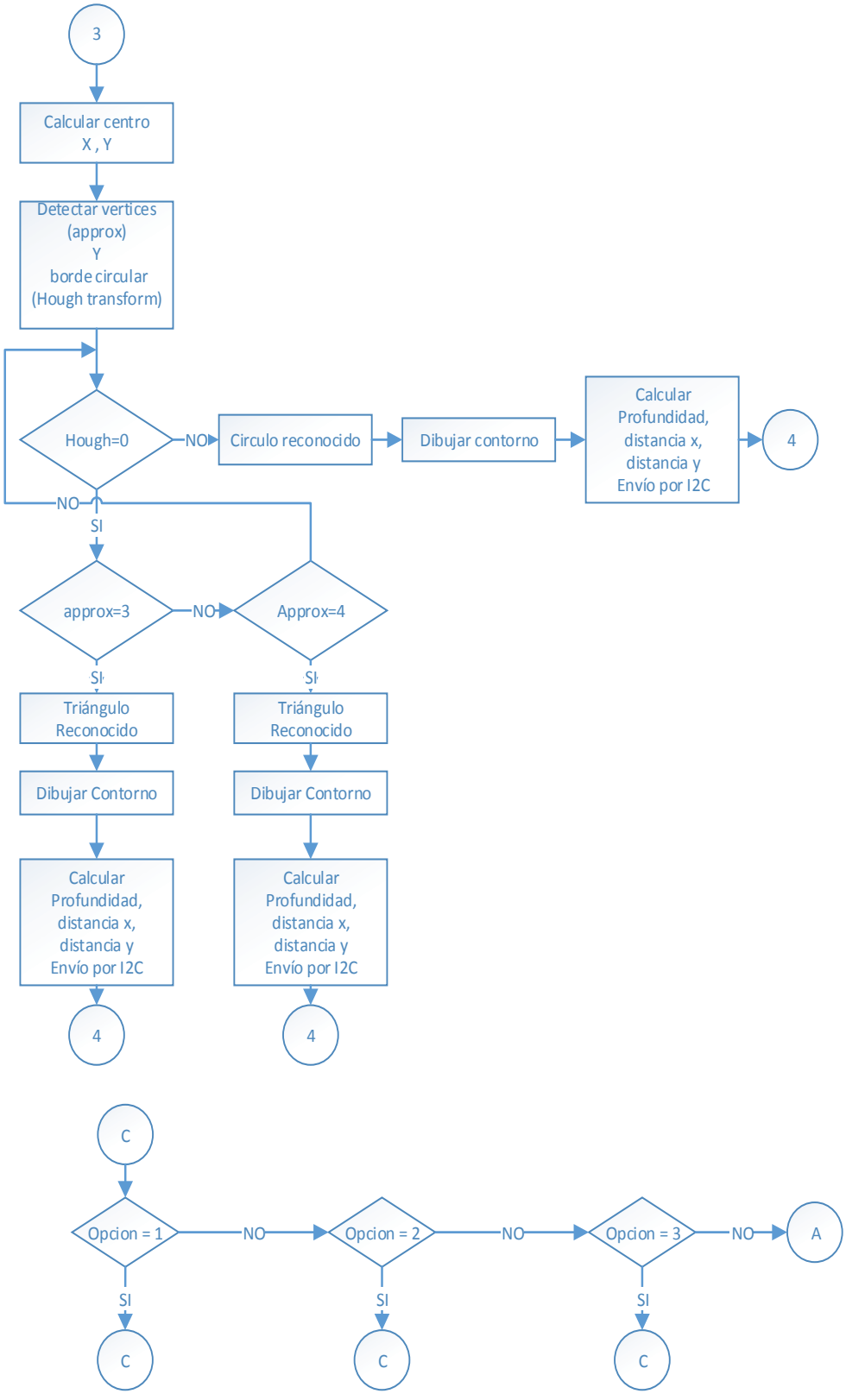


Figura 3.34. Programa principal, parte 2

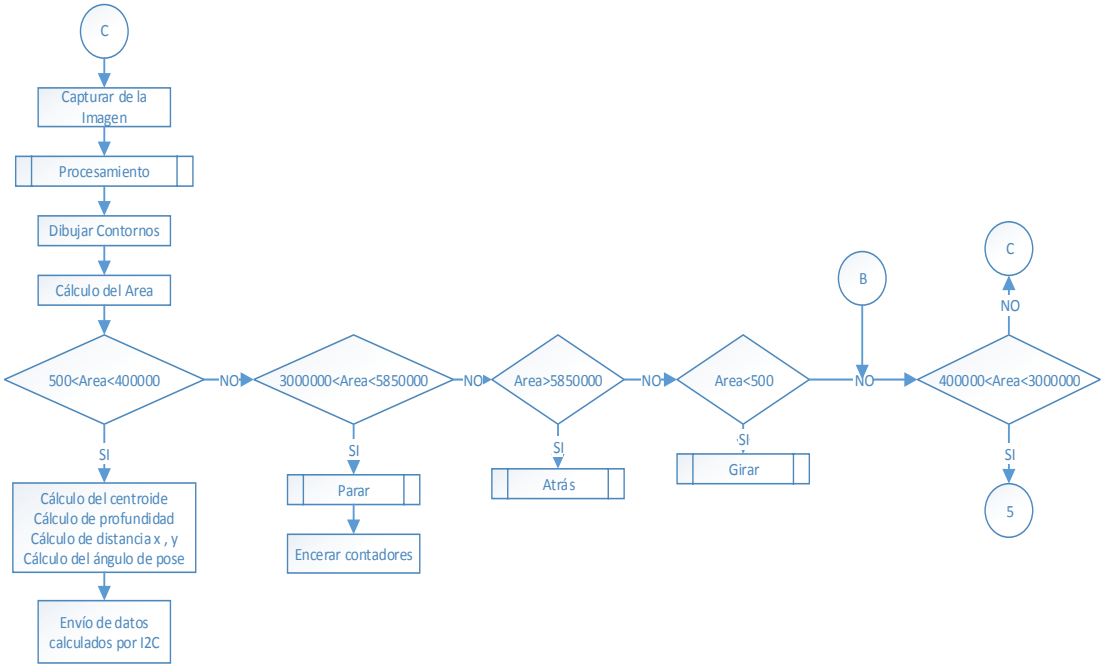


Figura 3.35. Programa principal, parte 3

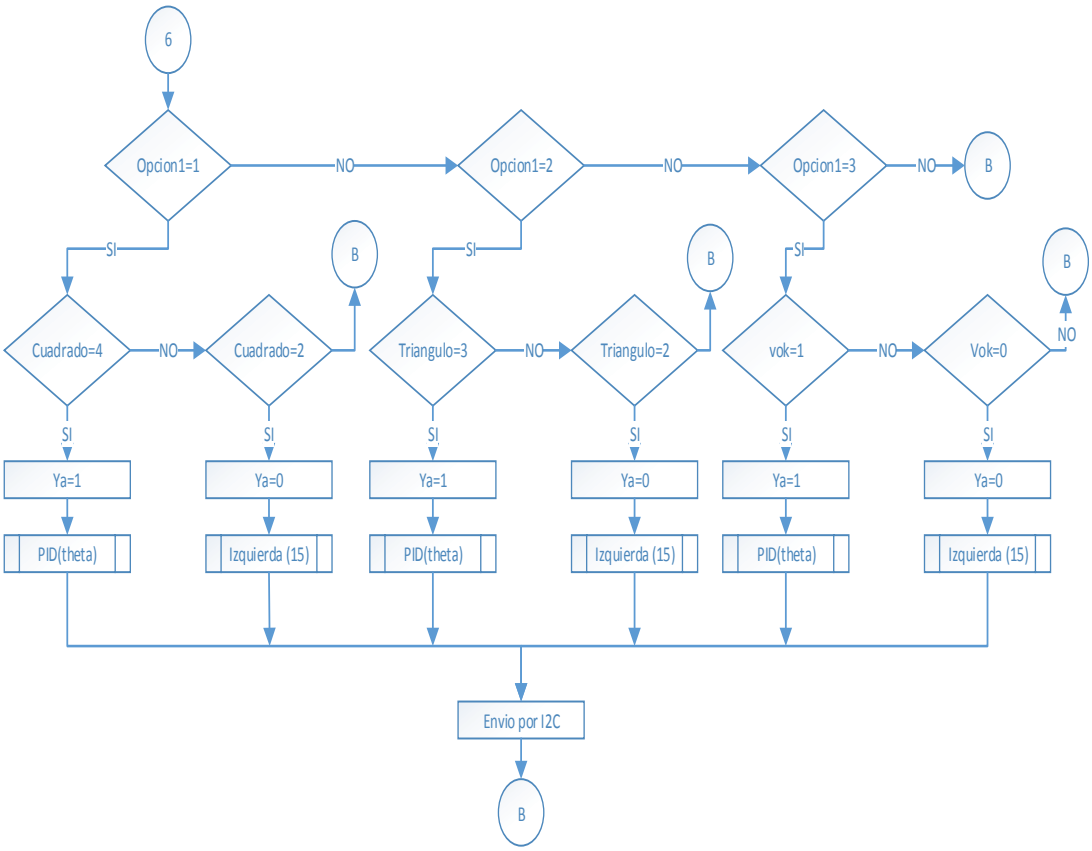


Figura 3.36. Programa principal, parte 4

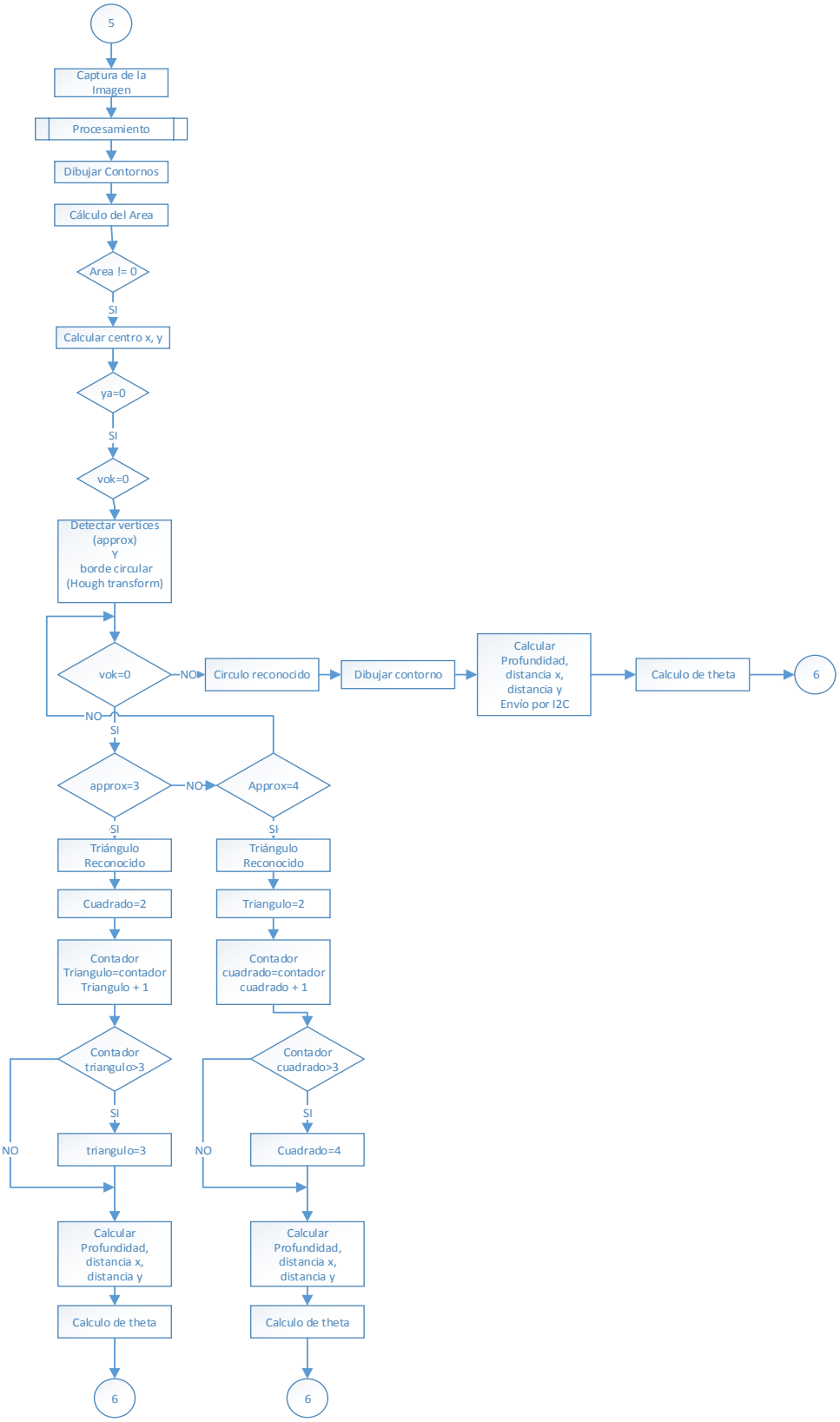


Figura 3.37. Programa principal, parte 5

3.6 CONCLUSIÓN DEL CAPÍTULO

Se logró realizar un algoritmo de detección de objetos predefinidos: rectángulo, triangulo y circulo, por su color y forma; para posteriormente poder realizar los cálculos que permitan determinar la posición del objeto reconocido y el robot pueda dirigirse a él.

CAPÍTULO 4

PRUEBAS Y RESULTADOS

En este proyecto se ha realizado el procesamiento de imagen por color y forma, en el cual se ha obtenido diferentes resultados que se detallarán en este capítulo.

4.1 NORMALIZACIÓN DE LA IMAGEN

Como se ha mencionado en el capítulo 1 sobre la segmentación a nivel de imágenes, los histogramas representan el número de pixeles para cada nivel de color de cada canal de manera independiente.

Ecualizar la imagen significa distribuir los niveles de color en todo el rango de la imagen de manera que no se concentre en un solo sitio de la misma.

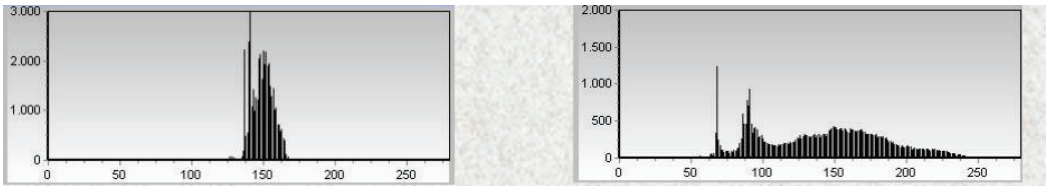


Figura 4.1. Ejemplo de ecualización de la imagen

Se realizaron dos pruebas basadas en la ecualización de la imagen original y la ecualización de la imagen en HSV que se detallarán a continuación.

4.1.1 ECUALIZACIÓN DE LA IMAGEN ORIGINAL

Cuando se trabaja con el procesamiento de imágenes generalmente se debe ecualizar los histogramas de cada canal, es decir, se separan los tres canales de la imagen, se ecualiza el histograma y se unen los tres canales ya ecualizados. En el presente proyecto se separó los canales BGR y se trabajó de manera independiente con cada canal, extrayendo el histograma y ecualizando el mismo, los resultados obtenidos son los siguientes.

Figura 4.2. a) Imagen original b) Imagen ecualizada

En la Figura 4.2 se observa que ecualizando la imagen los detalles de la misma son apreciados de mejor manera. Se transformó la imagen ecualizada a HSV y se realizó la respectiva umbralización con los valores de la Tabla 3.2 obteniendo el siguiente resultado:

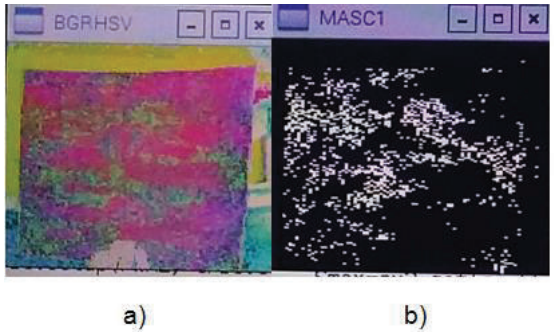


Figura 4.3. a) Imagen en HSV, obtenida a partir la imagen ecualizada previamente b) Umbralización, obtenida a partir de la imagen en HSV

En la Figura 4.3 no se tiene un resultado efectivo de la imagen detectada, debido a lo mencionado anteriormente. Como este proyecto no se basa en determinar los detalles de la imagen, sino solamente en la detección de color y forma, no se optó por la ecualización de la imagen original.

4.1.2 ECUALIZACIÓN DE LA IMAGEN HSV

Como la primera opción de ecualizar la imagen original no fue óptima, se realizó la ecualización en la imagen en HSV, es decir, la imagen original se transformó a HSV y la imagen transformada se ecualizó. Se realizó la respectiva umbralización según la Tabla 3.2 obteniendo el siguiente resultado:

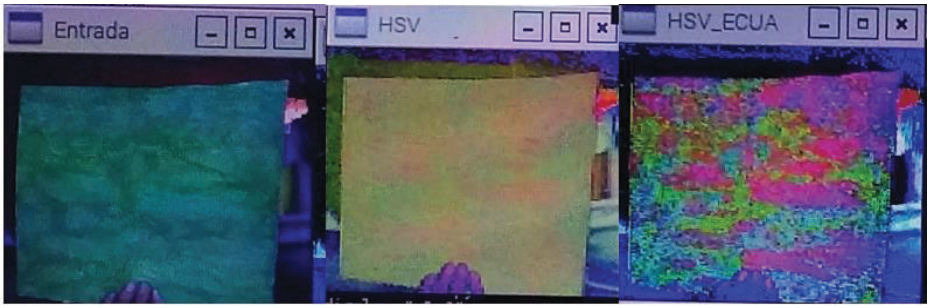


Figura 4.4. a) Imagen original b) Imagen en HSV c) Imagen HSV ecualizada

Como se puede observar en la Figura 4.5, se aprecian los detalles de la imagen y al aplicar la umbralización se obtiene el siguiente resultado:

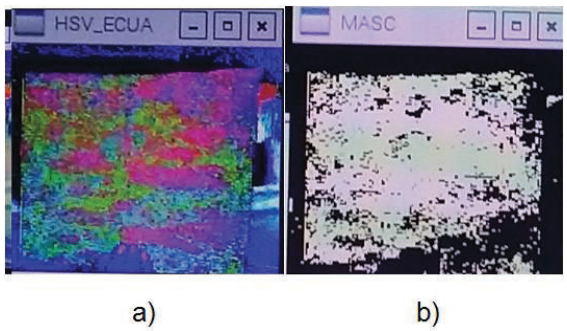


Figura 4.5. a) Imagen HSV ecualizada b) Umbralización de la imagen HSV

Como se observa en la Figura 4.5 se mejoró la umbralización de la detección por color a comparación del primer método (BGR), a pesar de ello no es suficiente, ya que con los filtros de cierre y erosión se tiene resultados erróneos. Por este motivo no se usó la normalización de la imagen, ya que no se obtiene una correcta segmentación.

4.2 CALIBRACIÓN CANALES HSV Y FILTRADO DE LA IMAGEN

Al trabajar con una imagen en modelo HSV se toma un rango de valores para cada canal, el cual se configura de acuerdo a la marca que se tenga, gracias a una función de opencv llamada “trackbar”, que permite modificar los valores de H-S-V sin salir del programa, esto facilita la detección de color.

De esta manera se obtuvo los valores correspondientes a la Tabla 3.2, detectando tres colores verde, azul y fucsia.

Las primeras pruebas se realizaron de la siguiente forma:

1. Captura de la imagen “Entrada”.

2. Realización Filtro de Media “imgMedian”.
3. Realización Filtro Gaussiano “GaussianBlur”.
4. Transformación a HSV “HSV”.
5. Realización de la segmentación según Tabla 3.2 “Mascara”.
6. Realización filtro morfológico de cierre “Closing”.
7. Realización de operación morfológica de erosión “Erode”

Se añadió ruido en la imagen para que se pueda observar de mejor manera cómo funcionan los filtros morfológicos utilizados.

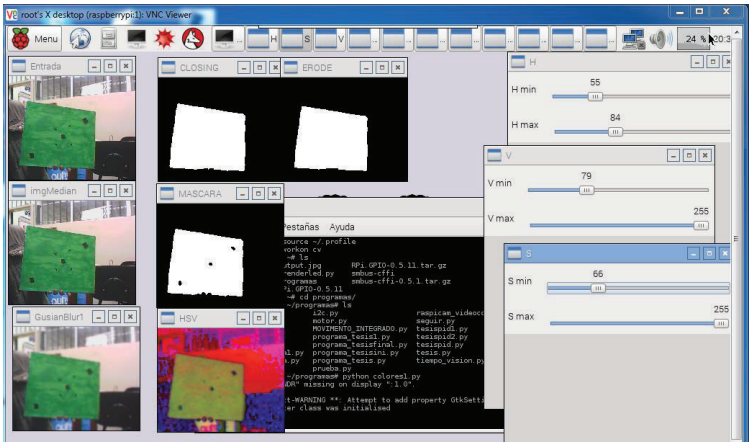


Figura 4.6. Procesamiento de imagen color verde.

Como se puede observar en la Figura 4.6, en el proceso de segmentación “Mascara” se puede identificar perfectamente el ruido añadido previamente en la imagen original “Entrada”. Con el filtro de cierre **“Closing” se elimina el ruido interno** de tal manera que al pasar al proceso de erosión elimina el ruido externo y el objeto es identificado.

Se realizó las mismas pruebas para el objeto de color azul y fucsia obteniendo los siguientes resultados:

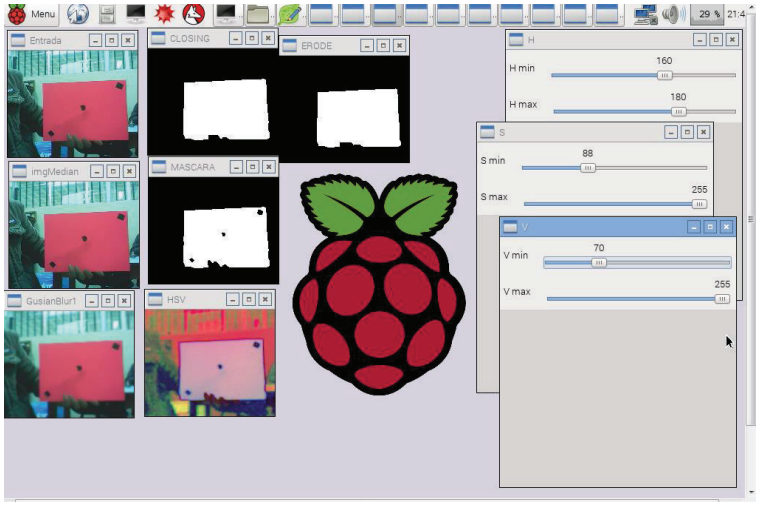


Figura 4.7. Procesamiento de imagen color Fucsia

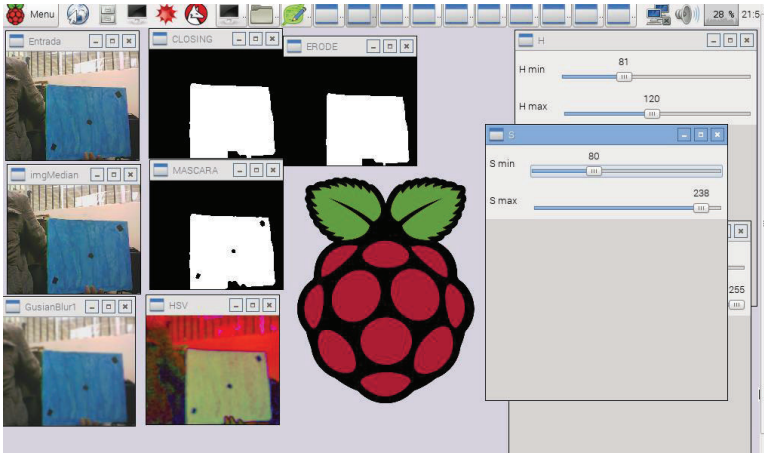


Figura 4.8. Procesamiento de imagen color azul

4.3 AMBIENTE NO CONTROLADO

Uno de los grandes problemas de la visión por computadora es el ambiente, ya que el mismo depende de factores como: iluminación, reflejo del objeto en el suelo, brillo, objetos en el entorno con las mismas características de las marcas, cantidad de luz, entre otros. Los principales problemas de detección que se ha tenido en el proyecto se deben al ambiente, ya que es un emisor de ruido aleatorio.

4.3.1 ILUMINACIÓN

La iluminación maneja un papel importante a la hora de detectar color, ya que el cambio de esta variable altera los parámetros de saturación, brillo y valor. El resultado obtenido es el siguiente:

Figura 4.9. Detección con poca iluminación

Como se puede observar en la Figura 4.9 con poca iluminación, la detección del verde y azul no son adecuadas, ya que esta variable genera cambios de las características de los mismos. Se puede aumentar el rango de saturación y brillo para solucionar este problema, pero esto conlleva tener errores en otro ambiente.

Figura 4.10. Detección con iluminación normal

Como se puede observar en la Figura 4.10 se detecta los tres colores con una iluminación normal.

4.3.2 REFLEJO DEL OBJETO EN EL SUELO

Uno de los principales errores en la detección es el reflejo de la imagen en suelos brillosos como baldosas, mármol, etc.

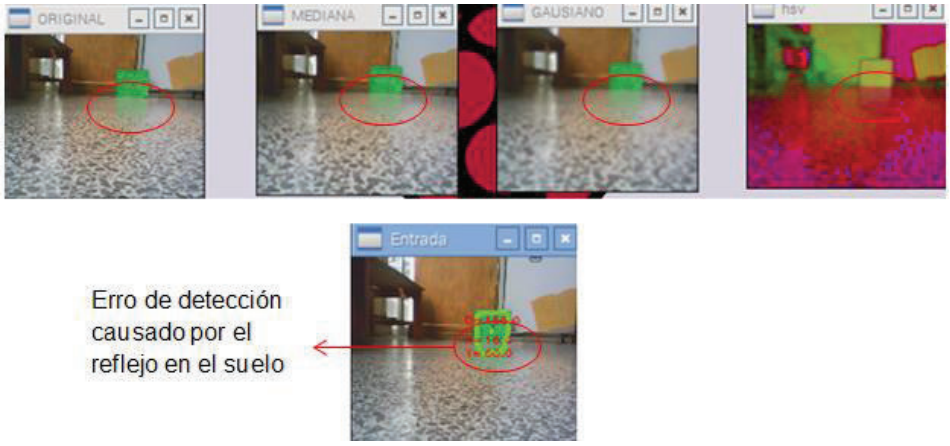


Figura 4.11. Error de detección causado por el brillo del suelo

Como se puede observar en la Figura 4.11, se tiene error en la detección causado por el brillo del suelo, lo cual afecta principalmente en la detección de contornos, ya que el número de vértices aumentan y no permite detectar correctamente la figura.

De la misma manera los reflejos afectan a todos los colores en este caso verde, azul y fucsia.

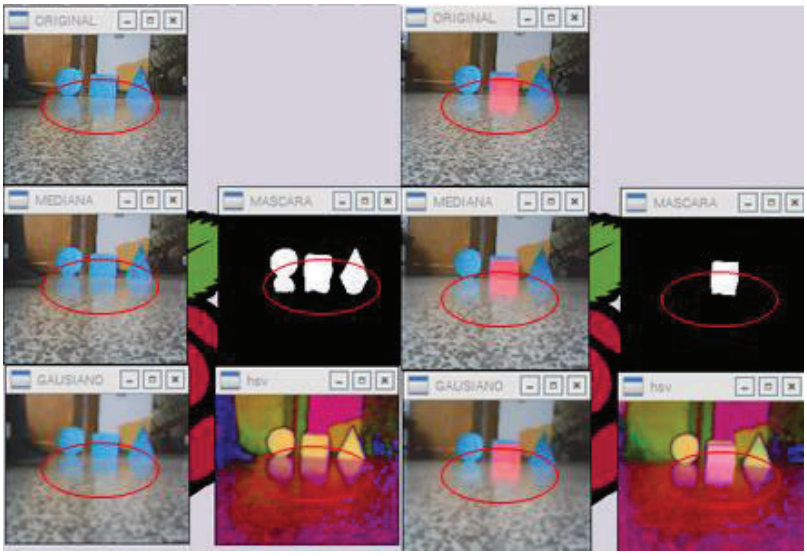


Figura 4.12. Error de detección color azul y fucsia por reflejo del suelo

Para lograr eliminar las falsas detecciones causado por el reflejo del suelo, se debe modificar la Tabla 3.2 aumentando la saturación, pero la desventaja de este método es que en sitios con baja iluminación no se puede detectar el color como

sucede en la Figura 4.9. De acuerdo a lo mencionado anteriormente, los nuevos valores de umbralización serán los siguientes:

Tabla 4.1. Ajuste de valores de umbralización

Color a detectar	H	S	V
Verde	Hmin=55 Hmax=85	Smin=98 Smax=255	Vmin=79 Vmax=255
Azul	Hmin=86 Hmax=120	Smin=125 Smax=238	Vmin=200 Vmax=255
Fucsia	Hmin=147 Hmax=180	Smin=100 Smax=255	Vmin=112 Vmax=255

De esta manera se corrigió los errores por reflejo del suelo obteniendo el siguiente resultado:

Figura 4.13. Detección de objetos con el ajuste de umbralización

4.3.3 CANTIDAD DE LUZ

Otro problema que se presenta en el proyecto es la cantidad de luz incidente al objeto a detectar, siendo esta condición más evidente cuando existen ventanas en el entorno.

Figura 4.14. Diferencia de cantidades de luz en un mismo entorno

En la Figura 4.14 se ubicó tres figuras con distinto color para observar la afectación en la detección. Las tres figuras por la cantidad de luz cambian los parámetros de color, saturación y brillo el cual no permite segmentar correctamente con los parámetros descritos en la Tabla 4.1.

Una posible solución es ajustar los parámetros de la Tabla 4.1, pero no es recomendable ya que se puede tener errores o falsas detecciones por el alto rango en la umbralización.

4.4 MEDICIÓN DE DISTANCIA

La medición de distancia se realiza en línea recta hacia el objeto a la que se ha denominado “profundidad”, en la desviación del objeto desde centro de la cámara hacia izquierda o derecha se ha denominado “x”, y en la medición de la altura a la que se encuentra el objeto se ha denominado “y”.

4.4.1 MEDICION DE PROFUNDIDAD

La medición de profundidad es realizada por medio de la semejanza entre triángulos el cual presenta errores para ciertos valores de distancia, por este motivo fue necesario crear una ecuación con el factor de corrección para cada medición, como puede observarse en la Tabla B.1.

Al aplicar la corrección se obtuvo el resultado mostrado en la Figura 4.15.

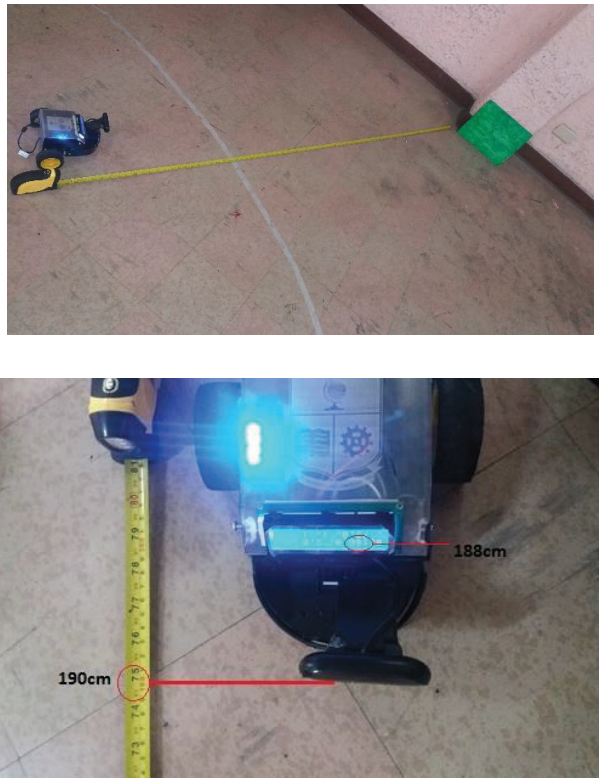


Figura 4.15. Medición de Profundidad

Las mediciones de profundidad a 4m presentan errores dentro del rango de ±10cm.

4.4.2 MEDICION ÁNGULO

La medición del ángulo presenta errores dentro de un rango de ±0.5° de desviación. Como puede observarse en la Figura 4.16, por lo que no requiere de un valor de ajuste.

Figura 4.16. Imagen con ángulo de desviación "A"

En la Figura 4.16 la centroide del objeto detectado se encuentra en el centro de la imagen por ende el ángulo de desviación es 0°. El valor del ángulo varía de -24° a 24° dependiendo de la ubicación del objeto.

4.4.3 MEDICION DE ALTURA (EJE Y)

Debido a la perspectiva de la cámara, es necesario colocar un valor de ajuste, a la medición de la altura del objeto respecto al suelo, el cual puede observarse en la Tabla B.2 en el Anexo B.

Cuyos resultados obtenidos se muestran en la Figura 4.17.

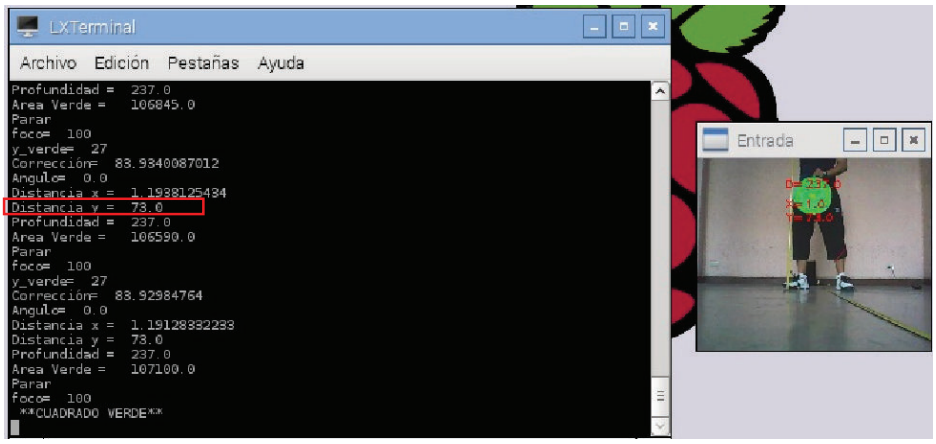


Figura 4.17. Medición de altura a 77 cm real.

El valor medido es de 73cm y el valor real es de 77cm. Las mediciones de altura presentan errores dentro del rango de ± 8 cm a una profundidad de 4mt.

4.5 AJUSTE CONTROLADOR DE POSICIÓN

El controlador de posición realizado es un PID el cual tiene los valores mostrados en el apartado 3.3, los cuales fueron calculados por el método de oscilación de Ziegler-Nichols

Se colocó un control proporcional Kp=Kc tal que el sistema empiece a tener oscilación continua como se muestra en la Figura 4.18.

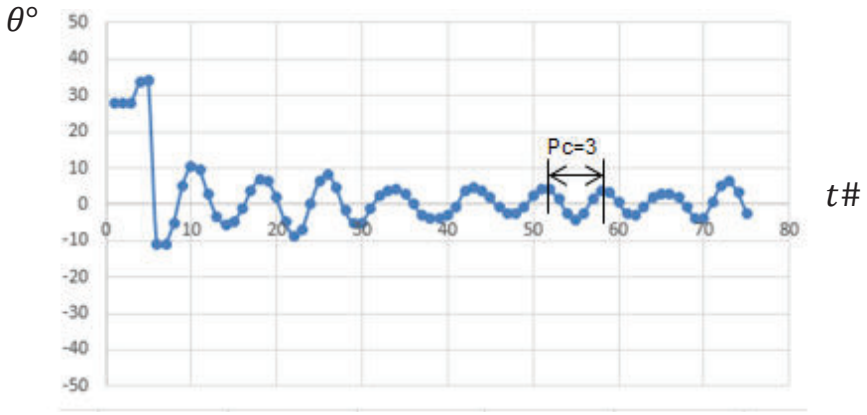


Figura 4.18. Sintonización del PID por Ziegler y Nichols con Kc=1,8, Kd=0 yKi=0

Como se muestra en la Figura 4.18 se tiene un Pc=3 y Kc=Kp=1,7 con ello se realiza los cálculos para obtener kp,ki y kd los cuales son los siguientes:

$$kp = 0{,}6 \cdot kc = 1{,}08 \tag{4.1}$$

$$Ti = 0{,}5 \cdot Pc = 1{,}5 \tag{4.2}$$

$$Td = \frac{Pc}{8} = 0{,}37 \tag{4.3}$$

$$Ki = \frac{1}{Ti} = 0{,}6 \tag{4.4}$$

$$Kd = Td = 0{,}37 \tag{4.5}$$

Se realiza un ajuste fino al controlador para mejorar la oscilación que aún se tenía al momento de acercarse al objeto. Los nuevos valores de Kp, Ki y Kd son los siguientes: K_p=1, K_d=0,3 y K_i=0,0001

El resultado obtenido se muestra en la Figura 4.19.

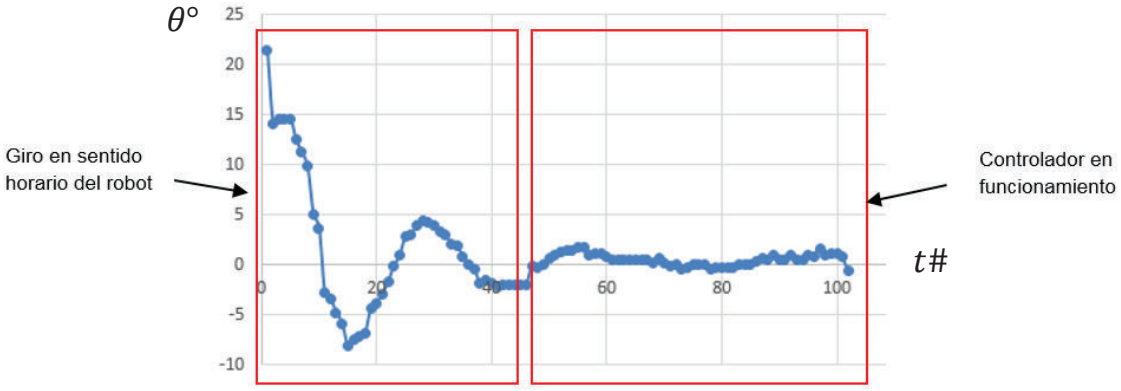


Figura 4.19. Controlador PID corregido

Los resultados obtenidos luego de la corrección realizada permiten al robot llegar a su objetivo sin perder el reconocimiento de la imagen.

4.6 COMUNICACIÓN I2C

Para la verificación de la comunicación I2C se utilizó un arduino nano con configuración esclavo cuya dirección es 0x07, el cual recibe el dato enviado por el maestro (raspberry pi) y lo muestra en un display 16x2.

El dato enviado corresponde a un vector que se conforma de la siguiente manera:

$$[profundidad, ángulo, altura, código\ color, código\ figura]$$

Entre los códigos de color y figura corresponde a la siguiente tabla:

Tabla 4.2. Códigos de color y figura enviados por I2C

Código Color	Color	Código Figura	Figura
1	Verde	1	Cuadrado
2	Azul	2	Triángulo
3	Fucsia	3	Circo

Se procede a tomar las formas de onda tanto de la línea SCL y SDA con la ayuda de un osciloscopio, para reforzar la teoría de la comunicación I2C el cual el resultado es el siguiente:

a) b)

Figura 4.20. a) Formas de onda SCA y SCL b) Dato enviado mostrado en display

En la Figura 4.20 se observa que las formas de onda de las líneas de comunicación corresponden a la teoría sobre el protocolo I2C explicadas en el capítulo 1. En el display se muestra el dato enviado por el maestro (raspberry pi), el cual se traduce de la siguiente manera: **el objeto se encuentra a 103cm de profundidad, con 1° de desviación, 40 cm de altura, es de color verde, y la figura es cuadrada**.

4.7 COSTO DEL PROYECTO

El objetivo del proyecto es minimizar los recursos económicos e implementar un hardware reducido en comparación a otros proyectos realizados, ya que al realizar visión por computadora generalmente se lo realiza en un computador o laptop de alta capacidad y costo, adicionalmente se ocupa mucho espacio físico al utilizar estas herramientas.

Tabla 4.3. Costos Tangibles del proyecto

COSTOS TANGIBLES				
	DESCRIPCION	**CANTIDAD**	**PRECIO UNITARIO**	**VALOR TOTAL**
1	Cámara Logictech c270	1	$ 66,10	$ 74,03
2	Raspberry Pi Modelo 2	1	$ 80,00	$ 89,60
3	Caja Raspberry Pi	1	$ 20,00	$ 22,40
4	Adaptador Wireless-USB	1	$ 20,00	$ 22,40
5	Power BANK	1	$ 32,00	$ 35,84
6	Driver Motores	1	$ 10,00	$ 11,20
7	Motores	2	$ 20,00	$ 44,80
8	Llantas	2	$ 8,00	$ 17,92
9	Bases de Sujeción	2	$ 3,00	$ 6,72
10	Rueda Loca	1	$ 3,00	$ 3,36
11	Juego de cables	1	$ 20,00	$ 22,40
12	Juego de Leds de alta intensidad	1	$ 5,00	$ 5,60
13	Juego de Baterías para motores	2	$ 10,00	$ 22,40
14	Interruptores	2	$ 0,50	$ 1,12
15	Arduino Nano	1	$ 15,00	$ 16,80
16	LCD	1	$ 7,00	$ 7,84
17	Potenciómetro	1	$ 0,25	$ 0,28
18	Estructura del robot	1	$ 80,00	$ 89,60
19	Pernos y Tuercas	1	$ 2,00	$ 2,24
20	Cables USB- Micro USB	2	$ 10,00	$ 22,40
	TOTAL			**$ 518,95**

COSTOS TANGIBLES				
	DESCRIPCION	**CANTIDAD**	**PRECIO UNITARIO**	**VALOR TOTAL**
21	Costo de Ingeniería	(400-Horas)	$ 20,00	$ 8.000,00
	TOTAL			**$ 8.000,00**

COSTOS TANGIBLES TOTALES				
	DESCRIPCION	**CANTIDAD**	**PRECIO UNITARIO**	**VALOR TOTAL**
1	Insumos	1	$ 518,95	$ 518,95
2	Costo de Ingeniería	1	$ 8.000,00	$ 8.000,00
	TOTAL			**$ 8.518,95**

En la Tabla 4.3 se divide en costos de insumos del proyecto y costos de ingeniería considerando que el proyecto práctico se realiza en 4 meses con dos personas. El valor total se consideró incluyendo el impuesto IVA.

Se realiza la comparación de costos de insumos con un proyecto similar llamado "Diseño e implementación de un prototipo a escala de un robot móvil acompañante" [46].

Tabla 4.4. Tabla comparativa de costos de insumos

Proyecto de Titulación	**Costos de insumos**
Diseño e implementación de un prototipo a escala de un robot móvil acompañante.	$1371,80
Diseño e implementación de un robot móvil provisto de visión artificial para reconocimiento de objetos y posicionamiento del móvil a los objetos	$ 518,95

Como se puede analizar en la Tabla 4.4, el presente proyecto es económico en insumos, independientemente del costo de ingeniería. Adicionalmente tiene poco espacio físico lo cual ayuda a la implementación.

4.8 CONCLUSIÓN DEL CAPÍTULO

Las pruebas realizadas muestran que los resultados obtenidos son los que se esperaba para poder realizar la detección de objetos en un sistema embebido, con los cuales se ha justificado la realización de este proyecto.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se demostró que es posible el uso de una SBC (Single Board Computer) de 32 bits, como la Raspberry Pi, para realizar el procesamiento de imágenes en línea, obteniendo resultados favorables en tiempo y costo.
- Se desarrolló un algoritmo que permita el cálculo de la distancia y posición de objetos conocidos a través de una cámara, por medio de la semejanza de triángulos encontrada en el modelo de la cámara pinhole, siendo necesario un factor de ajuste que permita tener precisión en la medición.
- El uso de software libre para el procesamiento de imágenes como son los programas OpenCV y Python han permitido realizar aplicaciones de visión por computador y control de dispositivos de una manera sencilla y rápida de entender, estas aplicaciones están solamente limitadas por las características físicas de memoria y procesamiento.
- Se desarrolló un algoritmo que permite reconocer objetos usando la técnica por color a una distancia de 4 m y permite al robot acercarse a los mismos a una distancia mínima de 30 cm, habiendo identificado previamente la forma del objeto que desea seguir el usuario.
- Se desarrollaron los algoritmos que permiten comunicar a la Raspberry Pi con Arduino por medio de comunicación I2C, tomando a la Raspberry Pi como maestro y a Arduino como esclavo, para posteriormente visualizar los datos enviados de profundidad, ángulo de pose, altura, el código de color y el código de objeto en un LCD colocado en la parte frontal del robot.
- En un ambiente no controlado los parámetros como el brillo, la cantidad de luz del ambiente, objetos de características similares al objeto a detectar, etc., influyen en el procesamiento de la imagen produciendo ruido, detección errónea o no detección del objeto que se quiere analizar.
- Debido a que la distribución del valor de cada color se realiza en una rueda de 360º, se descartó el color rojo ya que este se encuentra dividido entre

aproximadamente 330º a 360º y desde 0º a 20º lo cual no es posible representar en un rango de bits.

- Los filtros morfológicos son la mejor opción en la segmentación a nivel de objeto ya que nos permiten eliminar ruido externo como interno, gracias a las operaciones de dilatación, erosión, filtros open y close.
- Una imagen normalizada nos permite identificar los detalles que tiene una imagen ya sea para detección de ojos u otra aplicación, pero en este proyecto no se aplicó debido a que no se necesita detalles en la imagen sino solo color, adicionalmente se tiene varios errores en el proceso de segmentación.
- Se diseñó un robot tipo uniciclo con tracción diferencial mediante el programa inventor ya que este presenta diversas herramientas que ayudan al diseño del prototipo en 3D encajando perfectamente todos los dispositivos que conforman el mismo.
- Se elaboró un proyecto que tiene bajo costo en insumos de implementación, ya que se economiza al no utilizar un computador, sino una tarjeta computacional de bajo costo.
- Se realizó el algoritmo de medición de distancia, con el cual los resultados obtenidos presentan un error de 2.5% del valor máximo de profundidad, mismo que no afecta a la movilidad del robot y el reconocimiento de objetos por color.
- El algoritmo de medición de la altura se encuentra limitado a trabajar en un área que se encuentre al mismo nivel que el robot, es decir que la medida de altura partirá desde el nivel del suelo que se encuentre el robot.

5.2 RECOMENDACIONES

- Se debe realizar el ingreso a la Raspberry pi como usuario “root” ya que de esta manera se puede acceder a los archivos con permisos de super usuario, lo que permite controlar puertos de entrada y salida, creación de interfaz gráfica para visión por computador, protocolos de comunicación con otros dispositivos, etc.

- El uso de tarjetas micro USB de última generación como la clase 10, permite el acceso a datos con mayor velocidad, optimizando procesos complejos.
- Es importante recordar que la Raspberry Pi dentro de la comunicación I2C no puede trabajar en modo esclavo, solo puede trabajar como Maestro, por lo tanto la comunicación entre dos o más Raspberry Pi no es posible dentro de este protocolo de comunicación.
- La librería OpenCV presenta diferentes comandos que facilitan al procesamiento de la imagen, sin embargo se debe conocer de manera general la teoría que respalda a cada comando, ya que se puede cometer errores al colocar los parámetros sin estos conocimientos.
- Debido a la detección por color, que no permite tener dos objetos del mismo color en el plano de visión ya que el área detectada sería el conjunto de objetos que estén en el mismo plano, se recomienda realizar la detección de un solo objeto del mismo color a la vez.
- Con la finalidad de aumentar la velocidad de procesamiento de imágenes, se recomienda realizar un cambio de tamaño a la imagen. Esto permite disminuir el tamaño de las matrices que se resuelven y calculan para los diferentes filtros aplicados.
- La instalación de todos los paquetes y librerías necesarias para control de puertos de entrada y salida, comunicación I2C, visión por computador deben ser instalados dentro del directorio “root”, con la finalidad de que todo lo instalado posea permisos de super usuario.
- La conexión a la red formada por la Raspberry Pi y el computador que realiza el acceso remoto debe ser realizada por medio de un patch core proveniente del router al computador, ya que de lo contrario la imagen tomada por la cámara se demora en ser transmitida reduciendo la correcta visualización.
- Se recomienda que al ingresar por primera vez a la Raspberry pi se empiece a navegar por medio de la ventana de comandos llamada “*Terminal*” ya que esto desarrollará en el usuario habilidades de navegación dentro de carpetas y conocimiento de la distribución de archivos de configuración.

- Se recomienda leer el manual de usuario anexo, ya que en él se encuentra descrito el procedimiento correcto para la instalación de todos los paquetes y librerías con las versiones compatibles, necesarias para realizar este proyecto, además de la guía de manejo del robot.
- Se recomienda que el robot trabaje dentro de un rango de iluminación moderado.

REFERENCIAS BIBLIOGRÁFICAS

[1] Jose S. (2002) Avances en robótica y visión por computador. [Online]. https://books.google.com.ec/books?id=V-eXwzEIngQC&printsec=frontcover&dq=vision+por+computador&hl=es-419&sa=X&ved=0ahUKEwimsoLh6ojKAhXECpAKHXB2DRAQ6AEIGjAA#v=onepage&q=vision%20por%20computador&f=false

[2] J. P. Alvarado. (2013, Enero) Curso de Visión por Computador. [Online]. https://www.youtube.com/watch?v=sRXHGVRKrMk&index=1&list=PLsHprzbaic2_nJLfe9ZKVA9X-umEx7AY-

[3] A. Moreno, A. Sanchez y J. Sanchez J. Velez. (2003) Visión por Computador. [Online]. http://www.visionporcomputador.es/libroVision/libro.html

[4] Definicion Contraste, Saturación, Brillo y Matiz. [Online]. http://contenidos.educarex.es/mci/2002/24/actividades/6_9.html

[5] Fernando M. y Gianfranco P. (1995) Imagenes Medicas, adquisición, análisis, procesamiento e interpretación. [Online]. https://books.google.com.ec/books?id=Ah-qmV3wrs0C&pg=PA62&dq=histograma+de+imagenes&hl=es-419&sa=X&ved=0ahUKEwjhvt7Qh4nKAhUDhpAKHc92CKYQ6AEIGjAA#v=onepage&q&f=false

[6] A. Valero. (2013, Febrero) Principios del color y Holopintura. [Online]. https://books.google.com.ec/books?id=CXqrBAAAQBAJ&pg=PA147&dq=espacios+de+color&hl=es-419&sa=X&ved=0CBoQ6AEwAGoVChMIyomn-erWyAIVy6UeCh1kggQC#v=onepage&q=espacios%20de%20color&f=false

[7] Espacios de Color. [Online]. http://bibing.us.es/proyectos/abreproy/11875/fichero/Proyecto+Fin+de+Carrera%252F3.Espacios+de+color.pdf

[8] S. Folch. (2010, Mayo) Hablemos de comunicación multimedia. [Online]. http://www.comunicacion-multimedia.info/2010/05/modos-o-modelos-de-color-hsb-o-hsv-y.html

[9] Y. Gonzales. Imágenes en Color: Espacios de color. [Online]. http://dmi2.uib.es/~ygonzalez/Master_10212/Espacios_color_10210.pdf

[10] Python Software Fundation. (2015) Python. [Online]. https://www.python.org/

[11] M. Lutz. (2013, Junio) Learning Python. [Online]. https://books.google.com.ec/books?id=4pgQfXQvekcC&printsec=frontcover&dq=LEARNING+PYTHON&hl=es-419&sa=X&ved=0CBoQ6AEwAGoVChMIu8nDnM7YyAIVChGQCh2I3A0o#v=onepage&q&f=false

[12] A. Kaehler G. R. Bradski. (2008, Septiembre) Learning OpenCV. [Online]. http://www.cse.iitk.ac.in/users/vision/dipakmj/papers/OReilly%20Learning%20OpenCV.pdf

[13] T. Warner, *Hacking Raspberry Pi*, Primera ed., Greg Wiegand, Ed. United States of America: Que Publishing, 2013.

[14] Fundacion Raspberry Pi. (2015) Raspberry Pi - Productos. [Online]. https://www.raspberrypi.org/products/

[15] B. Horan. Google Drive (Practical Raspberry Pi). [Online]. https://drive.google.com/folderview?id=0B4HO-XVhuYRSR3IzREF2MDdvYXc&usp=sharing

[16] E. Upton. (2015, Febrero) BROADCOM. [Online]. http://www.broadcom.com/blog/raspberry-pi/love-to-get-your-hands-on-a-raspberry-pi-2-hat-tip-to-broadcom/

[17] J. García, X. Perramón J. Herrera. (2004, Julio) Aspectos avanzados en seguridad en redes. [Online]. https://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CBsQFjAAahUKEwiGr_i4v8fIAhWCg5AKHUyeB5Q&url=http%3A%2F%2Fwww.sw-computacion.f2s.com%2FLinux%2F012.1-Aspectos_avanzados_en_seguridad_en_redes_modulos.pdf&usg=AFQjCNFadAV

[18] Kutsuhiko O. (2003) Ingeniería de Control Moderna. [Online]. https://books.google.com.ec/books?id=QK148EPC_m0C&printsec=frontcover#v=onepage&q&f=false

[19] V. Mazzone. (2002, Marzo) Controladores PID. [Online]. http://www.eng.newcastle.edu.au/~jhb519/teaching/caut1/Apuntes/PID.pdf

[20] Gonzalo Z. (2007) ROBOTICA guía teórica y práctica. [Online]. https://books.google.com.ec/books?id=JPgyRgn-j1YC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[21] Victor G. (2002) Control Automatizado y Robotica. [Online]. http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/index.htm

[22] Mauricio R. y Santiago S. (2013, Sep) Diseño y construcción de tres mini robots exploradores cooperativos. [Online]. http://bibdigital.epn.edu.ec/bitstream/15000/6692/1/CD-5089.pdf

[23] Victor A. y Andrés R. Gabriela A. Modelación, Identificación y control de robots móviles. [Online]. http://bibdigital.epn.edu.ec/bitstream/15000/4912/1/Modelaci%C3%B3n,%20Identificaci%C3%B3n%20y%20Control%20de.pdf

[24] Juan G. (2003, Julio) Diseño de robots ápodos. [Online]. http://www.iearobotics.com/personal/juan/doctorado/cube-reloaded/download/tea.pdf

[25] Pololu Corporation. (2015, Noviembre) POLOLU. [Online]. https://www.pololu.com/product/3077

[26] Dome Pi. (2015, Mayo) Dome Pi Domotics. [Online]. http://domepidomotics.blogspot.com/2015/05/ventajas-y-desventajas-del-uso-de-la.html

[27] Logitech. (2015) Logitech. [Online]. http://www.logitech.com/assets/46920/2/webcam-c170-quickstart-guide.pdf

[28] LTD TP-LINK Technologies CO. (2015) TP-LINK User Guide TL-WR741ND. [Online]. http://www.tp-link.ec/res/down/doc/TL-WR741ND_V5_UG.pdf

[29] Samsung. Imagen Power Bank Samsung 9000 mAh. [Online]. https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcTxwDmfE_euDud1w83ijXvUbIKsHXUsu1V3873ykF3Vmff9wwMgGQ

[30] J. Mireles Jr. (2004, Feb) Kinematics of Mobile Robots. [Online]. http://www.uta.edu/utari/acs/jmireles/Robotics/KinematicsMobileRobots.pdf

[31] V. Andaluz, A. Rosales G. Andaluz. Modelación, Identificación y Control de robots móviles. [Online]. http://bibdigital.epn.edu.ec/bitstream/15000/4912/1/Modelaci%C3%B3n,%20Identificaci%C3%B3n%20y%20Control%20de.pdf

[32] Oya Y. and Richard E. Anne R. (2003, Feb) Moving Theory into Practice Digital Imaging Tutorial. [Online]. https://www.library.cornell.edu/preservation/tutorial-

spanish/tutorial_Spanish.pdf

[33] Francisco G. (2009, Nov.) Reconocimiento de objetos en una cocina con una webcam. [Online]. http://e-archivo.uc3m.es/bitstream/handle/10016/10007/PFC_FranciscoJavier_Garcia_Fernandez.pdf?sequence=3

[34] Fundación Open CV. (2015, Dec.) OpenCV 2.4.12.0 documentation. [Online]. http://docs.opencv.org/2.4/genindex.html

[35] Alexander M. and Abid K. (2014, Oct) OpenCv-Python Tutorials Documentation. [Online]. https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf

[36] Tosina J. Diseño de un filtro morfológico para la reducción de fluorescencia en espectros Raman. [Online]. https://upcommons.upc.edu/bitstream/handle/2099.1/6588/PFC_MARI_COMPLETO_definitivo.pdf?sequence=1&isAllowed=y

[37] Green B. (2002) Canny Edge Detection Tutorial. [Online]. http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/can_tut.html

[38] Princen J., Illingworth J. and Kittler J. Yuen H.K. (1989) A comparative study of Hough transform methods for cicle finding. [Online]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.386.8414&rep=rep1&type=pdf

[39] Martinez R. (2013, Mar) Detección de círculos con diferentes radios. [Online]. http://roberto-mtz.blogspot.com/2013/03/deteccion-de-circulos-con-diferente.html

[40] González D. Procesamiento de imágenes. [Online]. http://ocw.usal.es/eduCommons/ensenanzas-tecnicas/procesamiento-avanzado-de-imagenes-digitales/contenidos/Tema2.pdf

[41] Mukundan R. and Ramakrishnan KR. (1998, Sep.) Moment Functions in Image Analysis. [Online]. https://books.google.com.ec/books?id=2oDVCgAAQBAJ&printsec=frontcover&dq=moments+in+an+imagen+computer+vision&hl=es&sa=X&ved=0ahUKEwjajc7HktLJAhUEfZAKHWC3Ak8Q6AEIHDAA#v=onepage&q&f=false

[42] Pau L. and Wang P. Chen C. (1999, Mar) Handbook of pattern recognition and computer visión. [Online].

https://books.google.com.ec/books?id=8wvtCgAAQBAJ&printsec=frontcover&dq=moments+in+an+imagen+computer+vision&hl=es&sa=X&ved=0ahUKEwjajc7HktLJAhUEfZAKHWC3Ak8Q6AEIUTAH#v=onepage&q=moments&f=false

[43] Karina S. "Descriptores de imágenes digitales con momentos de Zernike", Master. Universidad Nacional Autónoma de México, Facultad de Ingeniería Eléctrica,2011. [Online]. http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/4682/tesis.pdf?sequence=1

[44] David B. (2015) PINHOLE.CZ. [Online]. http://www.pinhole.cz/en/pinholecameras/whatis.html

[45] Fernando S. "Taller de robótica y visión para estudios de grado", Master, Centro internacional de postgrado,Universidad de Oviedo, 2013. [Online]. http://digibuo.uniovi.es/dspace/bitstream/10651/18286/1/TFM%20-%20Fernando%20Soto.pdf

[46] López C Mesías A. (2012, Jun) Bliblioteca Digital EPN. [Online]. http://bibdigital.epn.edu.ec/bitstream/15000/4832/1/CD-4424.pdf

[47] DLINK. (2013, Octubre) Dlink wireless adapter manual. [Online]. ftp://ftp.dlinkla.com/pub/DWA-125/DWA-125_A3_Manual_v1.30(DI).pdf

[48] Dlink. (2013, Febrero) Manual de Usuario Dlink DIR600. [Online]. ftp://ftp.dlinkla.com/pub/DIR-600/DIR-600_B6_Manual_v6.02(ES).pdf

[49] ELECTRONILAB. Uso de Driver L298N. [Online]. http://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/

[50] S. Fernando. "Taller de Robótica y visión artificial para estudios de grado", Master,Centro internacional de Postgrado, Universidad de Oviedo, 2013. [Online]. http://digibuo.uniovi.es/dspace/bitstream/10651/18286/1/TFM%20-%20Fernando%20Soto.pdf

ANEXO A

A. MANUAL DE USUARIO

A.1 GUÍA DE INSTALACIÓN DEL SISTEMA OPERATIVO

Para iniciar con Raspberry Pi es necesario tener a mano los siguientes elementos:

Hardware

- Raspberry Pi
- Monitor con conector HDMI o en su defecto un conversor de HDMI a VGA.
- Mouse
- Teclado
- Cargador con conector micro USB de 2A.
- Adaptador Wireless con conector USB
- Cámara (Raspicam o Webcam), siendo la marca de cámaras web más compatible Logitech

Software

- SD Formatter
- Win32DiskImager
- Putty
- WNetWatcher

A.1.1 PREPARACIÓN DE LA TARJETA SD

El primer paso es formatear la tarjeta SD, para lo cual se utiliza el programa SD Formatter.

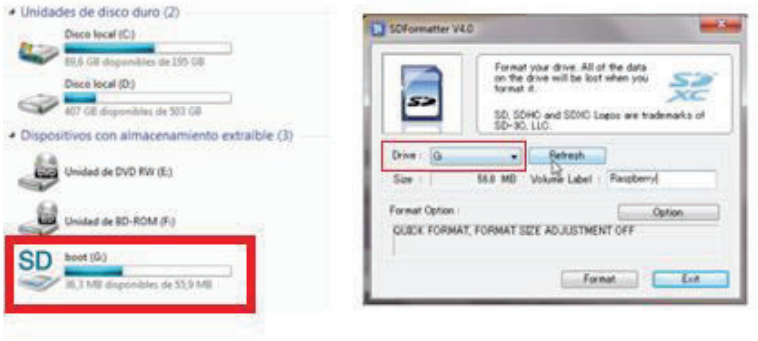


Figura A.1. Formateo de la tarjeta SD

Al ingresar la tarjeta SD al computador el programa automáticamente reconoce al dispositivo, es posible poner una etiqueta a la tarjeta para poder identificarla posteriormente.

En la pestaña opciones debe seleccionarse el tipo de formato (borrar todo) y el formato de ajuste de tamaño (activado).

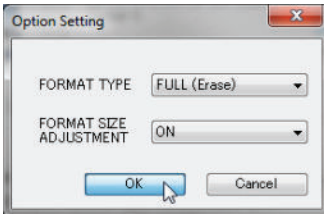


Figura A.2. Tipo de Formato de formateo

Posteriormente se da clic en Formatear y la tarjeta esta lista para poder grabar el sistema operativo.

A.1.1.1 Adquisición del sistema operativo

El sistema operativo recomendado por la fundación Raspberry pi es Raspbian, el cual puede ser adquirido ingresando a la siguiente dirección https://www.raspberrypi.org/downloads/, seleccionando la pestaña Raspbian se presenta la siguiente pantalla.

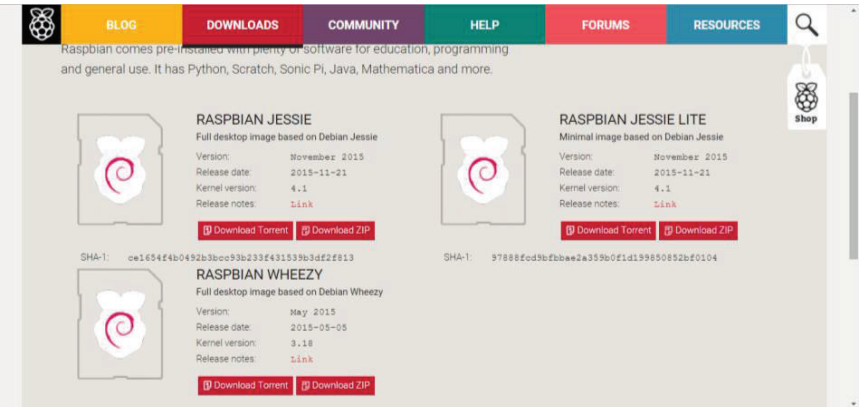


Figura A.3. Sistemas Operativos en la Web

La versión utilizada en este proyecto es RASPBIAN WHEEZY, el cual puede ser descargado por medio de Torrent o directamente como un archivo comprimido .ZIP, el cual deberá ser descomprimido obteniéndose un archivo .img

A.1.1.2 Grabar el sistema operativo en la tarjeta SD

Para poder grabar el sistema operativo raspbian wheezy en la tarjeta SD se hace uso del programa Win32DiskImager, donde se selecciona el icono en forma de carpeta para poder dar la dirección del archivo.img con el sistema operativo. En la pestaña “Device” se selecciona la etiqueta que representa la tarjeta SD en este caso la “G:”

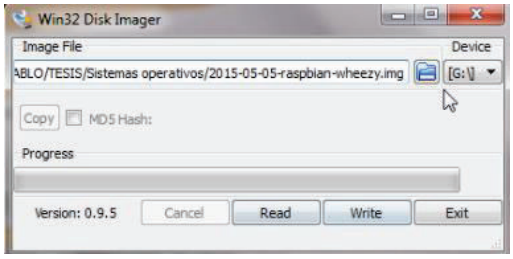


Figura A.4. Programa Win32DiskImager

Una vez seleccionado el archivo de imagen a ser grabado en la tarjeta SD, se debe dar clic en write. Tras esperar un tiempo el programa indicará que ya se ha realizado la grabación del archivo de imagen.

A.1.2 CONFIGURACIONES INICIALES DE LA RASPBERRY PI

Una vez colocada la tarjeta SD con el software grabado en la Raspberry Pi, lo primero que debe hacerse es ingresar al terminal. Para realizar configuraciones

preliminares que optimicen las funciones que presta la Raspberry Pi se utiliza el comando "*sudo raspi-config*", el cual muestra un menú con las siguientes opciones:

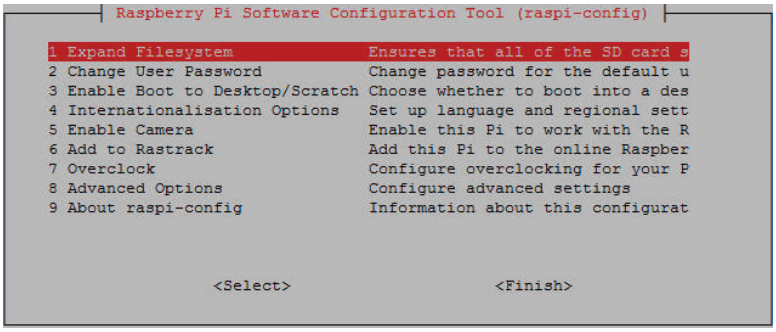


Figura A.5. Configuración del Sistema

A.1.2.1 Expand Fylesystem

Esta opción permite que la memoria destinada a almacenar todos los datos, librerías, paquetes, etc. Sea expandida al tamaño total de la tarjeta SD.

A.1.2.2 Change User Password

Esta opción permite cambiar la contraseña del usuario "*pi*" que se encuentra creado por defecto. La contraseña por defecto es "raspberry".

Es importante recordar que en Linux la escritura de la contraseña no es visible como medida de seguridad.

A.1.2.3 Enable Boot to Desktop/Scratch

Permite habilitar o deshabilitar el ingreso automático a la interfaz gráfica de Raspbian.

A.1.2.4 Internationalisation options

Permite realizar configuraciones como el idioma, zona horaria y tipo de teclado a usarse. Debe seleccionarse el idioma es_EC.UTF-8

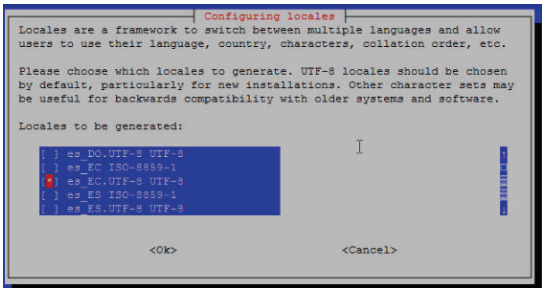


Figura A.6. Configuración local

La selección del idioma se hace por medio de la tecla “espacio”

A.1.2.5 Enable Camera

Permite habilitar o deshabilitar el uso de la Raspicam para la adquisición de imágenes o videos.

A.1.2.6 Overclock

Permite configurar la velocidad de procesamiento de la Raspberry Pi, el alterar este valor puede disminuir el tiempo de vida útil del dispositivo. Se recomienda de ser necesario aumentar este valor hasta “*Medium*”.

A.1.2.7 Advance options

Permite habilitar opciones de comunicación que se encuentran deshabilitadas por defecto.

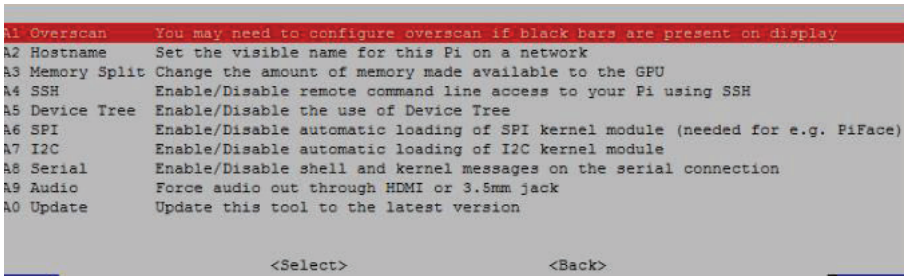


Figura A.7. Opciones Avanzadas

A.1.2 INGRESO COMO SUPER USUARIO

A.1.2.1 Super-usuario

Dentro de Linux y todas sus versiones se puede ingresar al sistema operativo como usuario por defecto “pi”, o como super-usuario “root”, el cual permite tener acceso a programas y archivos con todos los permisos como copia, escritura,

acceso a puertos de entrada y salida, etc. Es decir se tiene un control total de la Raspberry Pi.

A.1.2.2 Creación de super-usuario

Para poder ingresar como super-usuario es necesario primero ingresar como usuario "*pi*" y contraseña "raspberry".

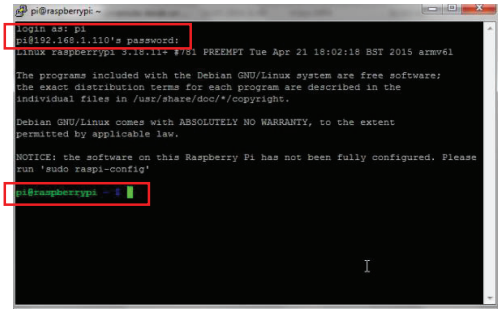


Figura A.8. Ingreso como usuario pi

Dentro del usuario pi se escribe el comando "sudo passwd root" con el cual se establece una contraseña para el super-usuario. Se pedirá confirmar la contraseña y tras dar un enter se observa un mensaje indicando que la contraseña fue creada satisfactoriamente.

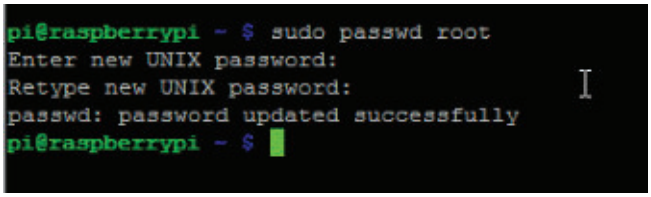


Figura A.9. Creación de usuario root

Tras reiniciar la Raspberry Pi con el comando "sudo reboot -h" debe ingresarse como usuario "root" y la contraseña antes creada.

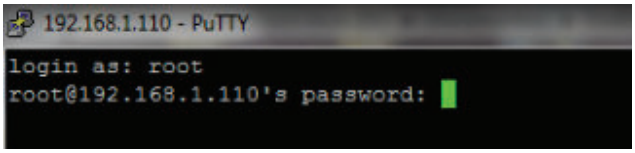


Figura A.10. Ingreso de clave y usuario root

A.2 CONFIGURACIÓN DE CONEXIÓN WIRELESS

La configuración de la conexión Wireless se realiza previa conexión del adaptador Wireless USB, posteriormente se debe ingresar a la Raspberry Pi como usuario "root", al realizar esto se evita escribir sudo en todos los comandos que necesiten permisos de super-usuario. Ingresando desde la ventana de comandos se digita la instrucción "iwlist scan" con la cual se muestran todas las redes que están dentro del rango disponible por el adaptador Wireless. Se debe copiar el nombre de la red que se encuentra listado como ESSID, en la imagen se observa ESSID: "TESIS VIRUS". También puede observarse que el puerto de red es el wlan0.

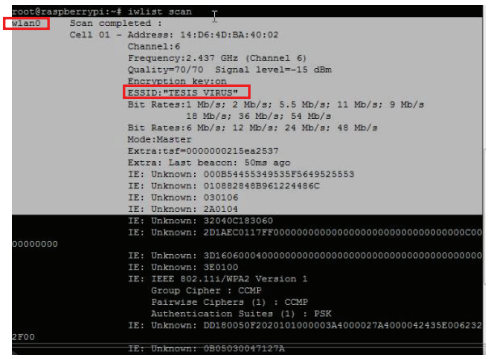


Figura A.11. Configuración de la Red Inalámbrica

Conociendo el puerto de red y el nombre de la red se procede a digitar el siguiente comando "iwlist wlan0 scan" con el cual se confirma que sea ése el puerto por el que se reconoce la red.

Para agregar el nombre de red y la contraseña de acceso a la misma se debe acceder al archivo de configuración *wpa-supplicant*. Por medio de la siguiente línea de comandos "*nano /etc/wpa_supplicant/wpa_supplicant.conf*".

Es necesario agregar el nombre y contraseña de la red a la que queremos ingresar. Quedando el archivo de la siguiente manera

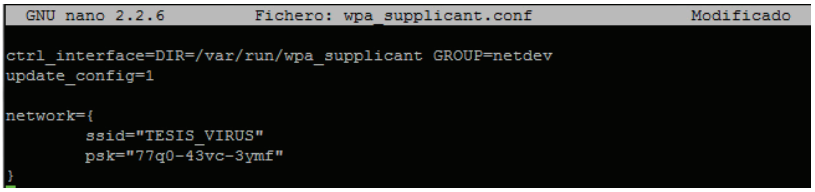


Figura A.12. Fichero wpa_suplicant

Se presiona "*ctrl+x*" para guardar el archivo y "*s*" para aceptar, finalmente se presiona "*enter*" y ya se ha configurado el acceso a la red.

Con la finalidad de que la Raspberry Pi reconozca automáticamente el puerto de conexión Wireless, ethernet o ambos a la vez se debe configurar el archivo "*interfaces*", al cual se accede por medio del siguiente comando "*nano /etc/network/interfaces*".

En los puertos de conexión que se desee que sean reconocidos automáticamente se debe cambiar la interface de manual a dhcp. Quedando el archivo de la siguiente manera.

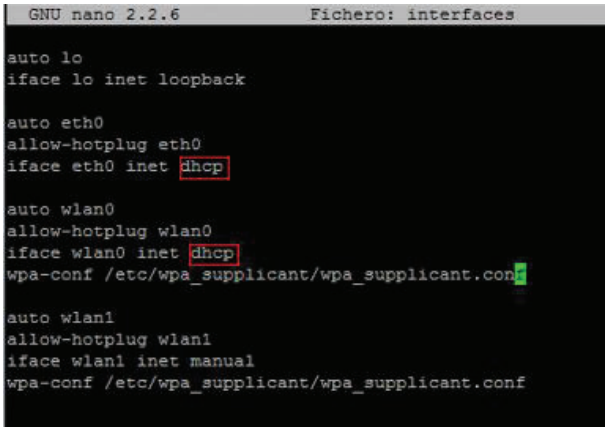


Figura A.13. Fichero interfaces

Se presiona "*ctrl+x*" para guardar el archivo y "*s*" para aceptar, finalmente se presiona "*enter*" y ya se ha configurado el reconocimiento automático del puerto de conexión de red.

A.3 ACCESO REMOTO A LA RASPBERRY PI

Otra forma de utilizar la raspberry pi es por medo de acceso remoto, este permite ya no conectar hardware externo como: teclado y mouse, sino realizarlo a través de la red con un computador ya sea de escritorio o laptop.

Figura A.14. Raspberry modo acceso remoto

Como se puede observar el la Figura A.14, los dispositivos indispensables para realizar el acceso remoto son el router, como administrador de la red, y los dos dispositivos tanto laptop como robot se pueden comunicar por este medio.

De esta manera se ahorra recursos de procesamiento, ya que la raspberry pi no está pendiente de procesar el mouse y teclado, todo esto es posible debido a que el sistema operativo raspbian tiene instalado previamente un protocolo SSH (Secure Shell), y gracias al escritorio virtual VNC (Virtual Network Computer).

A.3.1 PROCEDMIENTO PARA INGRESAR CON ACCESO REMOTO

Se tiene que entender que existen dos mundos, uno de ellos es el cliente y el otro es el servidor; en el caso de la raspberry pi se debe instalar un software para que se convierta en servidor, y en el computador de escritorio o laptop otro software para que se conviertan en clientes y puedan conectarse al servidor que es la raspberry pi.

A.3.1.1 Software para cliente (Ordenador de escritorio o Laptop)

1. Putty.- Este software se descargará de la página oficial de putty (link: http://www.chiark.greenend.org.uk/ sg-tatham/putty/download.html), el cual es un portal. El archivo es un ejecutable que ocupa poco espacio en el disco duro.

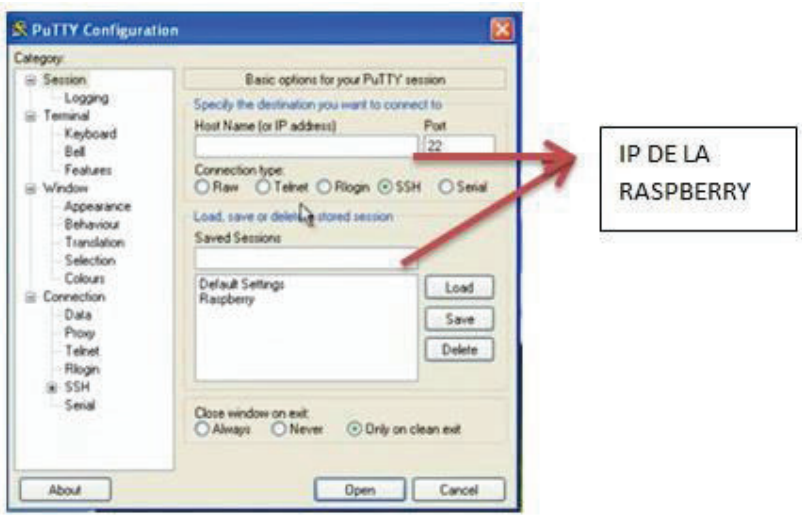


Figura A.15. Ventana de inicialización del Putty

En el interfaz putty se debe colocar en los dos espacios mencionados en la Figura A.15 la dirección IP que el router ha proporcionado a la raspberry, esto se podrá visualizar en la página oficial del router colocando en el navegador la siguiente dirección **192.168.0.1**. Por default la dirección asignada por el router a la rapsberry pi es **192.168.0.101** (No en todos los casos)**.**

Una vez ingresado la dirección IP en el Putty se debe colocar el usuario de la raspberry pi el cual es **root** y la clave que es **tesis1**, en ese instante ya se puede comandar el acceso remoto en la raspberry sin tener aún una interfaz gráfica, sino solamente por comandos desde nuestra laptop hacia la raspberry pi

2. VNC.- Este software es para establecer el escritorio remoto, se lo puede descargar desde la página oficial (link: http://www.realvnc.com/download/viewer/).
   Para poder visualizar la interfaz se ejecuta el **VNC** y se coloca la dirección IP de la Raspberry pi, separados por el carácter **(:)** la IP y el número de sesión como se muestra en la siguiente figura:

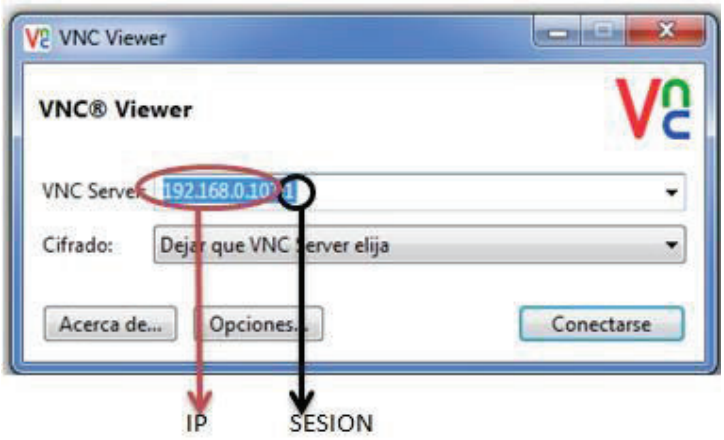


Figura A.16. Ventana VNC

Posteriormente se pedirá una clave la cual es **tesis1** para entrar al entorno virtual.

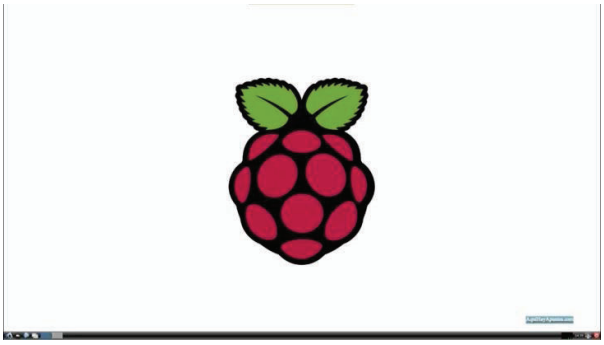


Figura A.17. Acceso remoto de interfaz con la raspberry pi

A.3.1.2 Software para servidor (Raspberry pi)

El software de instalación se recomienda que se haga desde el acceso remoto, ya configurado previamente por medio del putty. En la ventana del putty luego de haber ingresado la clave y usuario se escriben los siguientes comandos:

- apt-get update: actualiza los repositorios.
- apt-get update: actualiza los paquetes del Sistema Operativo.
- apt-get install tightvncserver: instala VNC server en la raspberry pi.
- Vnserver :1 –geometry 1366x768 –depth 24.- sirve para acceder al acceso de escritorio en el cual **1** representa los repositorios, el siguiente la geometría de la pantalla, que se recomienda que sea 1366x768 tamaño estándar.

NOTA: La primera vez que se ejecuta este comando se pedirá una clave, la misma que servirá para conectarse nuevamente en un futuro, se recomienda recordar la clave. En este proyecto la clave es **tesis1**. Luego se preguntará si se desea ejecutar el cliente solo como vista Y/N, se coloca N, porque se quiere interactuar con el servidor; esta pregunta es muy importante para el acceso de escritorio ya que si se coloca Y, se tendrá la interfaz gráfica sin los permisos necesarios para trabajar con visión por computador.

A.4 GUÍA DE INSTALACIÓN DE OPENCV Y PYTHON

Las versiones instaladas para este proyecto son Python 2.7 y OpenCV 2.4.10, las cuales han sido compatibles dentro de la Raspberry Pi. Dentro de la ventana de comandos se debe actualizar y mejorar el sistema y los paquetes instalados, esto se logra con los comandos "*apt-get update*" y "*apt-get upgrade*".

Figura A.18. Ejemplo de aplicación del comando

Una vez actualizado el sistema de debe reiniciarlo, para posteriormente instalar todos los paquetes requeridos por Python y OpenCV, por medio del comando "*apt-get install build-essential cmake pkg-config*". Los paquetes build-essential y pkg-config se encuentran por defecto instalados, pero por si no lo estuvieran se recomienda instalarlos.

Ahora se instala los paquetes de imagen necesarios para cargar imágenes en diferentes formatos como JPEG, PNG, TIFF, etc. Esto se logra con el comando "*apt-get install libjpeg8-dev libtiff4-dev libjasper-dev libpng12-dev*"

Como siguiente paso se debe instalar la librería GTK que es la encargada de crear interfaces de usuario gráficas y es requerida por la librería de OpenCv denominada *highgui* la cual permite visualizar imágenes desde el entorno de OpenCV. Esto se puede realizar por medio del comando "*apt-get install libgtk2.0-dev*"

Se procede a instalar los paquetes necesarios para videos, por medio del comando "*apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev*"

Se instala las librerías que son necesarias para optimizar ciertos procesos en OpenCV, con el comando "*apt-get install libatlas-base-dev gfortran*".

Se instala el paquete "*pip*", el cual permite instalar en adelante ciertas librerías para OpenCV, por medio de los siguientes comandos "*wget https://bootstrap.pypa.io/get-pip.py*" y "*python get-pip.py*"

Se instala el paquete que permite crear el entorno virtual donde trabajará OpenCV. Por medio de los siguientes comandos, uno después de otro "*pip install virtualenv virtualenvwrapper*" y "*rm -rf ~/.cache/pip*".

Ahora se procede a abrir el archivo ~/.profile, si no se encuentra puede ser creado, pero dentro de la carpeta root. Esto se realiza con el siguiente comando "nano ~/.profile"

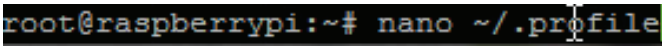


Figura A.19. Ubicación del fichero profile

Una vez abierto o creado se observará la siguiente pantalla

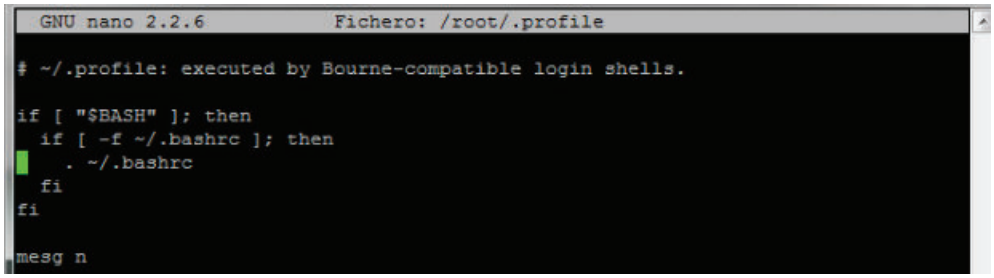


Figura A.20. Archivo de texto profile

Dentro de este archivo debe adjuntarse las siguientes líneas al final

 # virtualenv and virtualenvwrapper

export WORKON_HOME=$HOME/.virtualenvs

source /usr/local/bin/virtualenvwrapper.sh

Quedando el archivo como se observa en la siguiente imagen

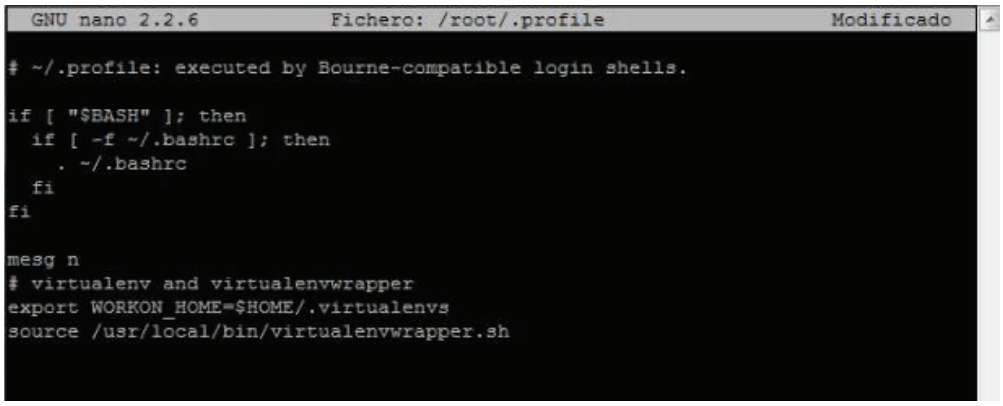


Figura A.21. Cambios del archivo profile

Se presiona "*ctrl+x*" para guardar el archivo y "*s*" para aceptar, finalmente se presiona "*enter*" y con esto se ha creado el archivo ~/.profile.

Se debe recargarlo con el comando "*source ~/.profile*".

Figura A.22. Cargar el archivo profile

Ahora se creará el entorno virtual de visión por computador con el comando "*mkvirtualenv cv*".

Figura A.23. Creación virtual para trabajar con Opencv

Es importante observar que aparezcan las letras "CV" en el indicador del usuario en los comandos.

Figura A.24. Aplicación del comando para trabajar con opencv

De ahora en adelante cualquier programa o paquete que necesite trabajar conjuntamente con OpenCV debe ser instalado previamente ingresando a este entorno virtual "CV".

Ahora se instalará Python en su versión 2.7, por medio del comando "*pt-get install python2.7-dev*" y su herramienta de trabajo con matrices numpy por medio del comando "*pip install numpy*"

Una vez instalado Python se procede con la instalación de OpenCV versión 2.4.10. Opencv puede ser descargado de la página oficial por medio del comando "*wget -O opencv-2.4.10.zip http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.10/opencv-2.4.10.zip/download*".

El archivo descargado se guarda dentro de la carpeta root, esto es importante ya que en esta carpeta se tiene permisos de superusuario. Por lo que se procede a descomprimir el archivo con el siguiente comando "*unzip opencv-2.4.10.zip*" y se abré la carpeta creada con el comando "*cd opencv-2.4.10*"

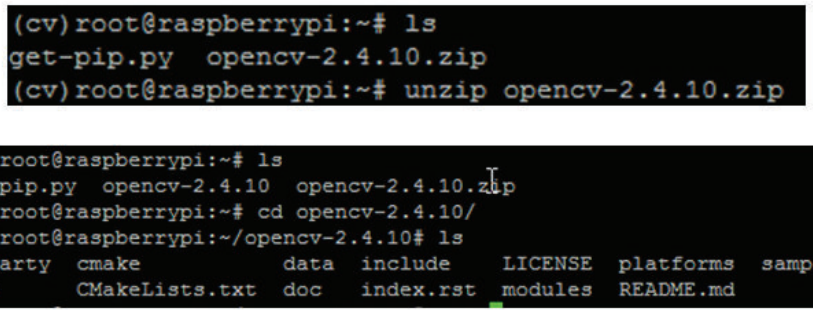


Figura A.25. Captura de pantalla donde se encuentra opencv

Se debe crear una carpeta desde la cual se ejecutará el instalador de OpenCV con el comando "*mkdir build*", se ingresa a la carpeta por medio del comando "*cd build*" y se construye los archivos de instalación con el comando "*cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_NEW_PYTHON_SUPPORT=ON -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON ..*".

Una vez terminado y dentro de la carpeta build se procede a compilar OpenCV por medio del comando "*make*"

Figura A.26. Aplicación del make dentro del directorio

Este proceso dura aproximadamente 2.5 horas en el modelo Pi 2. Una vez terminado se procede a instalar OpenCV por medio de los comandos "*make install*" y "`ldconfig`".

Ahora ya se ha instalado OpenCV el cual debe encontrarse en la siguiente dirección "*/usr/local/lib/python2.7/site-packages*".

Aunque para poder utilizar OpenCV dentro del entorno virtual "CV" se debe simular OpenCV dentro del directorio "*site-packages*". Esto se realiza por medio de los siguientes comandos.

```
cd ~/.virtualenvs/cv/lib/python2.7/site-packages/

ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so

ln -s /usr/local/lib/python2.7/site-packages/cv.py cv.py
```

Finalmente se comprueba que la instalación de OpenCV este correcta.

```
(cv)root@raspberrypi:~/.virtualenvs/cv/lib/python2.7/site-packages# workon cv
(cv)root@raspberrypi:~/.virtualenvs/cv/lib/python2.7/site-packages# python
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'2.4.10'
>>>
```

Figura A.27. Comprobación de instalación Opencv

A.5 GUÍA DE INSTALACIÓN DE LIBRERÍAS GPIO Y SMBUS

A.5.1 INSTALACIÓN DE LIBRERÍA GPIO

La librería GPIO permite tener acceso a los pines de entrada y salida disponibles en la Raspberry Pi, sin embargo se necesita permisos de super usuario para trabajar, por lo que es importante asegurarse que antes de la instalación se ingrese a la carpeta "root" y se haya activado el entorno virtual "CV", esto puede realizarse por medio de los siguientes comandos "*source ~/.profile*" y "workon cv"

```
root@raspberrypi:~# source ~/.profile
root@raspberrypi:~# workon cv
(cv)root@raspberrypi:~# python prenderled.py
```

Figura A.28. Inicialización del terminal

Primeramente se descarga la librería Rpi-Gpio desde la página oficial https://pypi.python.org/pypi/RPi.GPIO

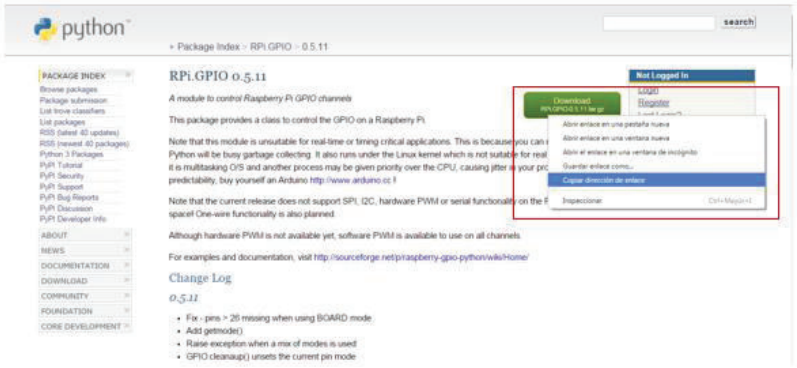


Figura A.29. Captura de pantalla link de descarga python

Se copia la dirección de enlace y se coloca en la ventana terminal con el siguiente comando “wget dirección_enlace”

```
(cv)root@raspberrypi:~# wget https://pypi.python.org/packages/source/R/RPi.GPIO/R
Pi.GPIO-0.5.11.tar.gz
```

Figura A.30. Comando de instalación GPIO

El archivo descargado puede ser visualizado con el comando “ls”, se puede observar que el archivo es descargado en formato .tar

```
(cv)root@raspberrypi:~# ls
get-pip.py  opencv-2.4.10  opencv-2.4.10.zip  output.jpg  RPi.GPIO-0.5.11.tar.gz
```

Figura A.31. Lista de descargas

Para descomprimir el archivo se coloca el siguiente comando “*tar –zxvf nombre del archivo*”

Ahora se ingresa a la carpeta descomprimida con el comando “*cd nombre_del_archivo*”.

```
(cv)root@raspberrypi:~# ls
get-pip.py      opencv-2.4.10.zip  RPi.GPIO-0.5.11
opencv-2.4.10   output.jpg         RPi.GPIO-0.5.11.tar.gz
(cv)root@raspberrypi:~# cd RPi.GPIO-0.5.11/
(cv)root@raspberrypi:~/RPi.GPIO-0.5.11# ls
CHANGELOG.txt  LICENCE.txt  README.txt  setup.py  test
INSTALL.txt    PKG-INFO     RPi         source
```

Figura A.32. Ingreso del archivo descargado

Dentro de la carpeta se debe ejecutar el archivo “*setup.py*” por medio del siguiente comando “*python2.7 setup.py install*”

Con esto queda instalada la librería GPIO.

A.5.1 ACTIVACION DE COMUNICACIÓN I2C E INSTALACIÓN DE LIBRERÍA SMBUS

Para activar la comunicación I2C previamente se debe instalar la librería Smbus y herramientas que permitan controlar la comunicación esto se realiza por medio de los comandos “apt-get install python-smbus” y “apt-get install i2c-tools”

Una vez instaladas las librerías necesarias, se debe activar la comunicación i2c en las configuraciones de la Raspberry Pi, a las que se accede por medio del comando “raspi-config”.

Se selecciona Advance Options

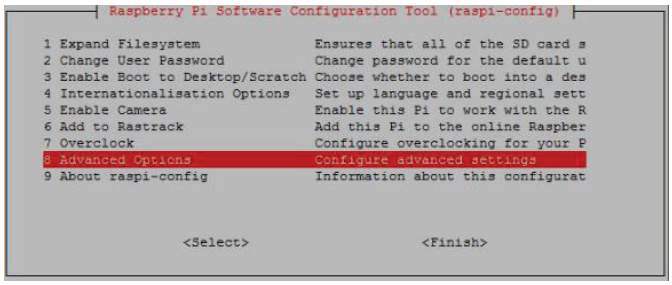


Figura A.33. Ingreso configuración I2C

Se debe seleccionar la comunicación i2c presionando “enter”

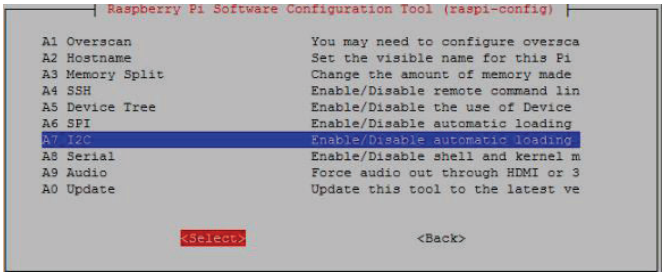


Figura A.34. Habilitar interface I2C

Se pregunta si se gustaría habilitar la interface I2C, aceptamos

Se acepta la habilitación de la interfaz y se muestra un mensaje indicando que la interfaz se activara después del reinicio de la Raspberry Pi.

Al aceptar se observa la pregunta si se desea que se cargue el kernel i2c por defecto.

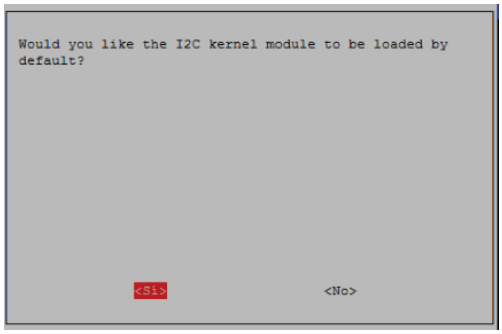


Figura A.35. Habilitación del Kernel

Se acepta la pregunta anterior y se sale de la interfaz.

Se procede a reiniciar la Raspberry Pi con el comando “reboot -h”.

Al encender nuevamente la Raspberry se debe realizar la instalación del kernel i2c manualmente. Para esto se ingresa al archivo modules con el siguiente comando “nano /etc/modules” y se agrega las siguientes líneas al final del archivo.

i2c-bcm2708

i2c-dev

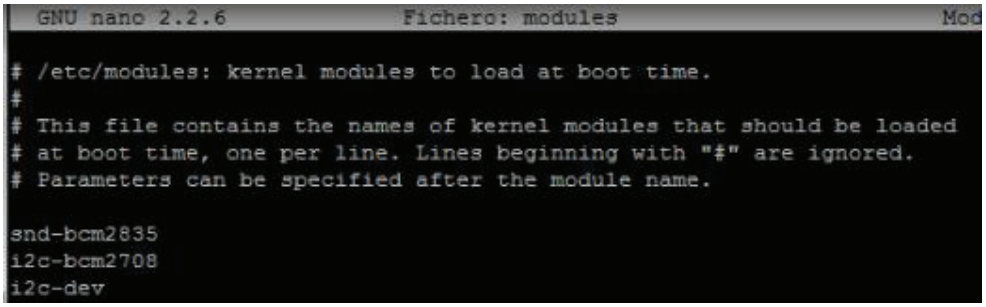


Figura A.36. Habilitación de la librería I2C

Una vez guardado el archivo. Dependiendo de la versión de Raspberry Pi se puede tener o no el archivo raspi-blacklist.conf en la dirección /etc/modprobe.d/raspi-blacklist.conf el cual debe ser editado. Comentando las siguientes líneas con un “#” en frente de cada una de ellas. Quedando el archivo de la siguiente manera.

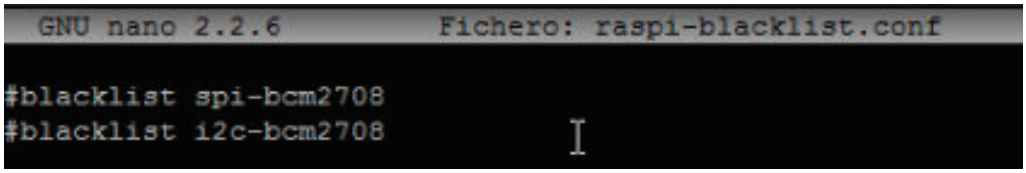


Figura A.37. Fichero raspi-blacklist

También es necesario actualizar el archivo config.txt que se encuentra en la dirección /boot/config.txt, agregando las siguientes lineas al final del archivo

dtparam=i2c1=on

dtparam=i2c_arm=on

Se reinicia la Raspberry Pi y al encenderla se comprueba que funcione la comunicación i2c, habiendo conectado previamente el dispositivo esclavo, por medio del comando “i2c detect –y 1”.

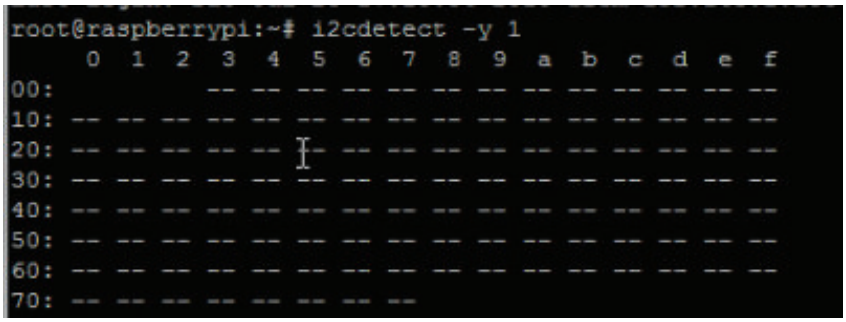


Figura A.38. Verificación de esclavos conectados

En la imagen no se ha conectado ningún dispositivo a la Raspberry Pi, por lo que no se observa ninguna dirección en la tabla.

A.6 GUIA DE MANEJO DEL ROBOT

A.6.1 ENCENDIDO DEL ROBOT

Es importante recordar que primero debe encenderse el router ya que este debe reconocer a la Raspberry Pi para asignarle una dirección IP.

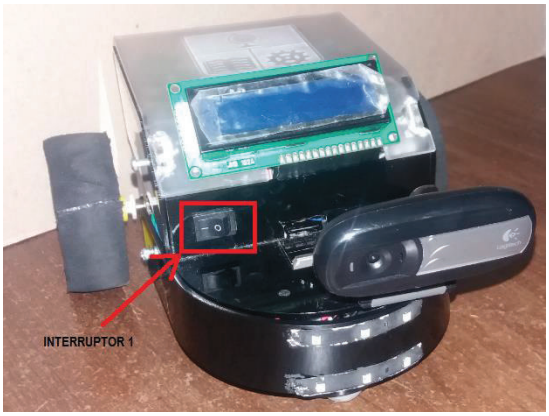


Figura A.39. Vista frontal del Robot

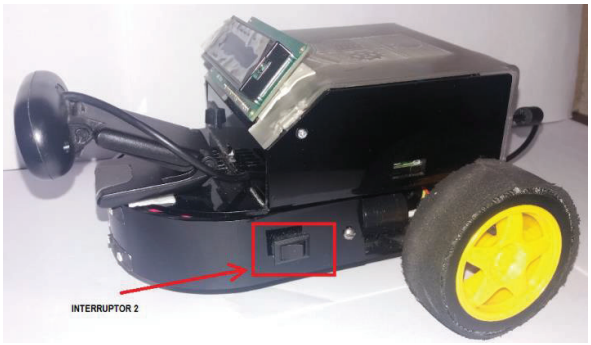


Figura A.40. Vista lateral izquierda del robot

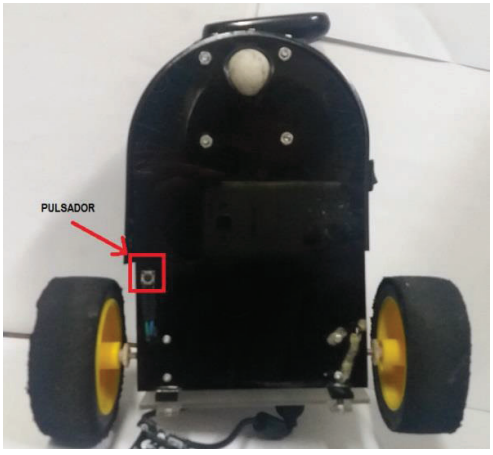


Figura A.41. Vista inferior del robot

El encendido del robot se realiza por medio de 2 interruptores y un pulsador.

Se empieza presionando el pulsador, cuya posición se observa en la Figura A.41, con el cual se enciende el banco de baterías que alimentan a las tarjetas de control.

Seguidamente se presiona el interruptor 1, cuya posición puede observarse en la Figura A.39, con el cual se enciende la Raspberry Pi.

A continuación se presiona el interruptor 2, cuya posición puede observarse en la Figura A.40, con el cual se enciende el driver que distribuye la energía a los motores.

A.6.1.1 Ingreso por acceso remoto

Una vez encendida la Raspberry Pi se debe ingresar a su interfaz gráfica por medio de acceso remoto.

Se inicia revisando la dirección IP asignada a la Raspberry Pi por el router. Para el efecto en un explorador de internet ingresamos la dirección “192.168.0.1”, inicia una pantalla donde se ingresa el nombre de usuario “*admin*” y la contraseña “admin”.

Figura A.42. Ingreso clave router

Al ingresar a la configuración del router en el lado inferior derecho de la pantalla se selecciona la pestaña DHCP, y dentro de esta la pestaña DHCP Clients List, donde se observa los dispositivos conectados a la red con sus respectivas IP asignadas.

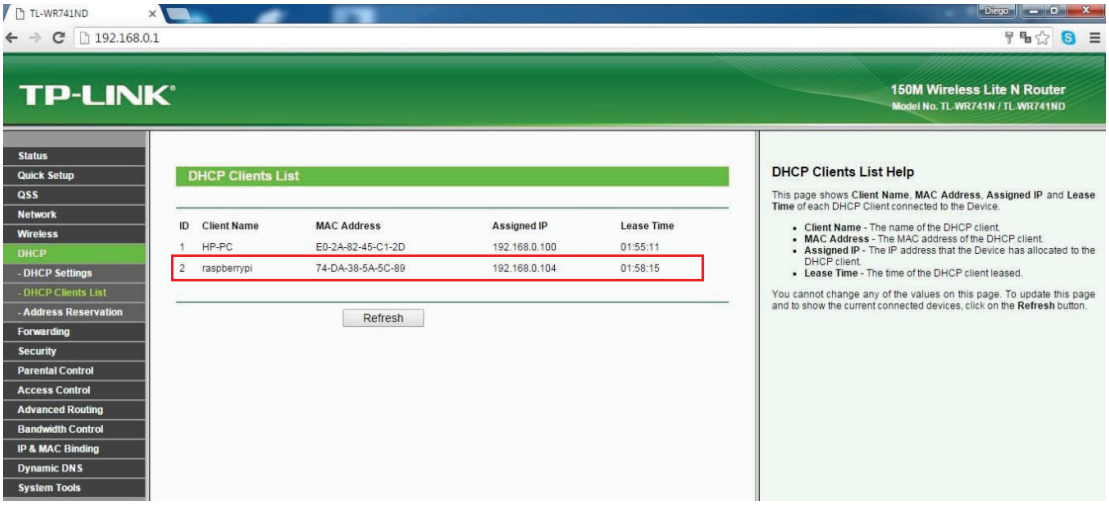


Figura A.43. Visualización IP raspberry pi

Una vez conocida la dirección IP se procede a ingresarla en el programa Putty, el cual crea una interfaz de acceso remoto por comandos.

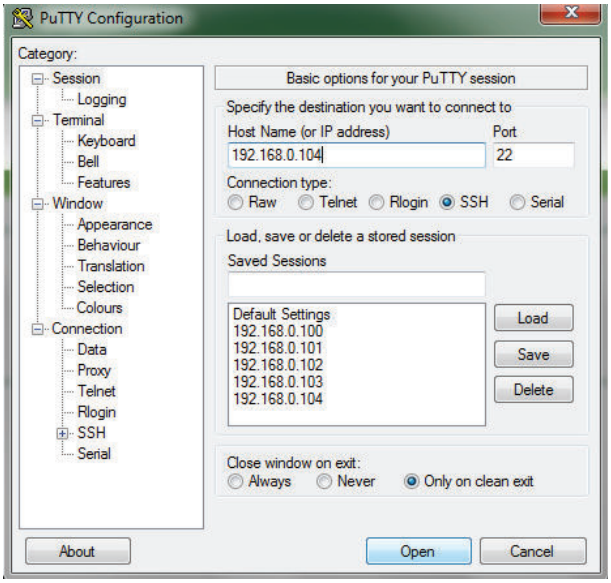


Figura A.44. Ventana Putty

Dentro de la ventana de comandos Digitamos el nombre de usuario “root” y la contraseña “tesis1”

Figura A.45. Ingreso al putty

Una vez dentro de la Raspberry Pi se debe activar la interfaz gráfica por medio del comando “tightvncserver”, con el cual se muestra que se ha creado una sesión y se indica su número.

```
(cv)root@raspberrypi:~# tightvncserver
```

Figura A.46. Comando para inicializar entorno virtual

Posteriormente se inicia el programa VNC, en el cual se ingresa la dirección IP de la Raspberry Pi, se muestra una ventana que indica que la conexión no estará cifrada, se presiona continuar.

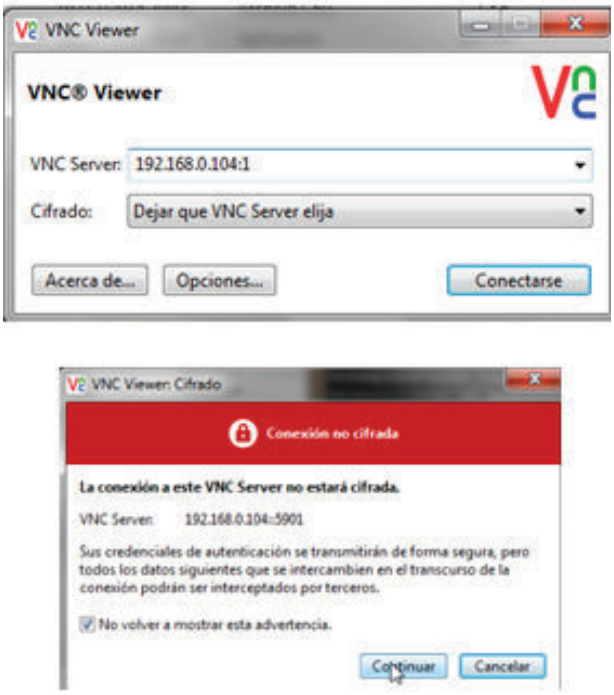


Figura A.47. Ventana VNC

Posteriormente se muestra otra pantalla en la que se ingresa la contraseña “tesis1”

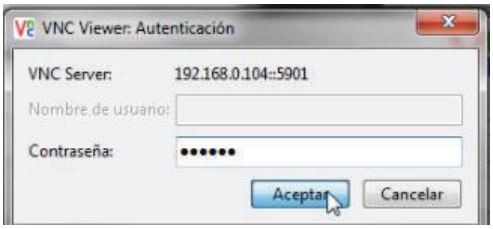


Figura A.48. Ingreso Clave VNC

Al presionar Aceptar se ingresa al escritorio remoto creado, donde se procede a abrir la ventana del terminal de comandos. Desde el cual se ejecutará el programa del robot móvil.

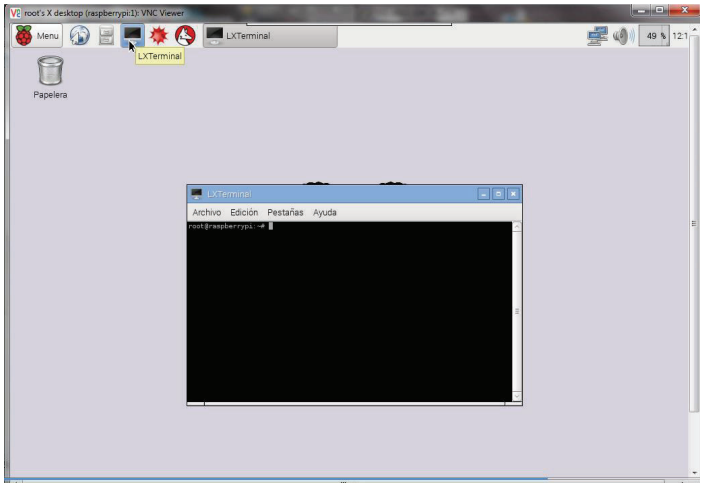


Figura A.49. Pantalla de inicialización raspberry pi

A.6.1.2 Activación de interfaz virtual (CV)

Para activar la interfaz virtual se utiliza los siguientes comandos “source ~/.profile” y “workon cv”.

```
root@raspberrypi:~# source ~/.profile
root@raspberrypi:~# workon cv
```

Figura A.50. Inicialización en el ternimal

A.6.1.3 Ingreso al programa principal

El programa principal se encuentra dentro de la siguiente dirección /root/programas/tesispid2.py

Como se ingresó previamente como usuario root, basta con colocar los siguientes comandos para acceder a la dirección antes mostrada.

Se digita “*cd programas*” para poder acceder a la carpeta programas dentro del usuario root

Se digita “*ls*” para poder visualizar el listado de documentos dentro de la carpeta programas, en el cual se ubica el programa principal llamado “*tesispid2.py*”.

Para ejecutar el programa principal se utiliza el programa Python, por lo que el comando de ejecución sería “*python tesispid2.py*”.

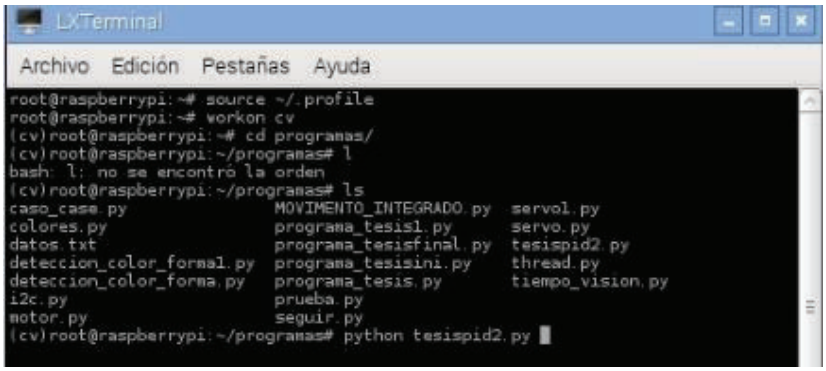


Figura A.51. Comandos de ingreso al programa de detección

A.6.2 MANEJO DEL PROGRAMA

El programa inicia creando la ventana de visualización de la cámara y muestra los datos calculados en la ventana terminal.

El programa en primera instancia hace que el robot gire en sentido horario mostrando por pantalla todo lo observado, si reconoce algún objeto mostrará sus coordenadas (theta, y), la distancia a la que se encuentra del robot, la forma de la figura (cuadrado, triangulo o circulo) y su color (verde, azul o fucsia) por medio de códigos vistos la Tabla 4.2.

El usuario dispone de tres teclas de control por teclado, “a”, “r”, “q”. Cuyas funciones se describen a continuación

- **Tecla “q”:** permite al usuario salir del programa en cualquier momento.
- **Tecla “a”:** permite al usuario entrar al modo de reconocimiento y seguimiento. En pantalla se presentará la pregunta “Elija la opción de color del objeto que desea seguir 1.- VERDE 2.-AZUL 3.- AMARILLO” y la

pregunta “Elija la opción de forma del objeto que desea seguir 1.- CUADRADO 2.-TRIANGULO 3.- CIRCULO”.

- **Tecla “r”:** permite al usuario regresar al estado inicial del programa, es decir, salir del modo de reconocimiento.

Nota: Es necesario que el cursor se encuentre sobre sobre la ventana de la imagen y no sobre el LXterminal, ya que se ha ingresado la pregunta se coloca el cursor en el LXterminal para escribir el tipo de reconocimiento que se prefiera.

Al inicio se realiza la calibración de la saturación mínima dependiendo del entorno al cual se va a realizar la detección, visualizando la ventana de umbralización llamada “Mascara” de cada color, hasta que la detección se considere correcta y no afecta cambio de luz o brillo en el suelo.

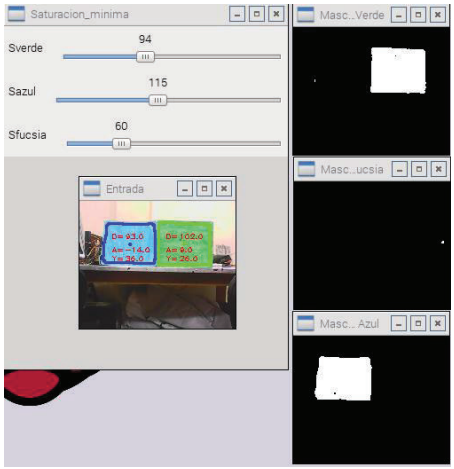


Figura A.52. Calibración Saturación mínima para cada color

A.6.3 APAGADO DEL ROBOT

Para apagar el robot debe escribirse en la ventana de comandos “init 0”, lo que apagará la Raspberry Pi, una vez apagada la Raspberry Pi se apaga el interruptor 1, y se procede a apagar el banco de baterías manteniendo presionado el pulsador en la parte inferior del robot por más de 3 segundos.

Finalmente se desconecta la energía del driver de los motores por medio del interruptor 2.

A.6.4 CARGA DE BATERIAS

A.6.4.1 Carga del banco de baterías para el control

La carga del banco de baterías del control se realiza por medio de un cargador de 2A, con conector micro USB. El cual debe ser conectado tal como se muestra en la figura.

La tarjeta mostrará la carga de las baterías representándola por medio de 4 leds azules. Los cuales se encenderán dependiendo de la carga. Mientras se continúen cargando las baterías el led siguiente parpadeará.

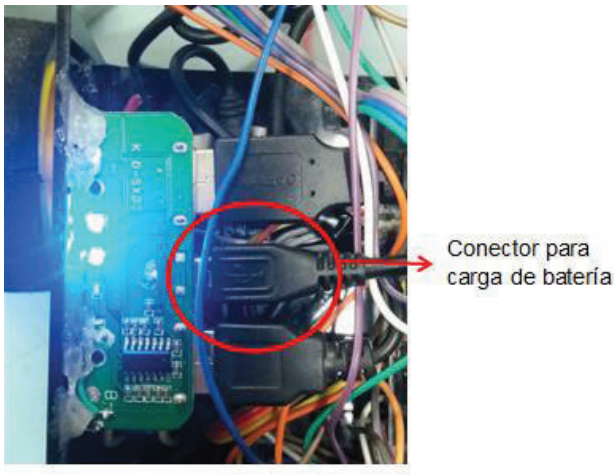


Figura A.53. Pin de Conexión carga de baterías raspberry pi

A.6.4.2 Carga del banco de baterías para los motores

Para realizar la carga de baterías de los motores se debe separar la parte superior de la inferior del robot, en el frente se encuentran dos conectores hembra para poder realizar la carga. Siendo el cable negro positivo y el blanco negativo.

ANEXO B

B. TABLAS DE CORRECIÓN DE DISTANCIAS

B.1 CORRECCIÓN DE PROFUNDIDAD

Tabla B.1. Corrección de Profundidad

PROFUNDIDAD		
Distancia	**Valor Medido**	**Corrección**
30	27	3
40	30	10
50	38	12
60	45	15
70	52	18
80	63	17
90	72	18
100	81	19
110	87	23
120	96	24
130	106	24
140	111	29
150	122	28
160	132	28
170	140	30
180	148	32
190	157	33
200	167	33
210	178	32
220	188	32
230	197	33
240	212	28
250	218	32
260	232	28
270	247	23
280	258	22
290	267	23
300	282	18
310	292	18
320	306	14
330	327	3
340	335	5
350	352	-2

360	366	-6
370	383	-13
380	399	-19
390	408	-18
400	438	-38

B.2 CORRECCIÓN DE ALTURA

Tabla B.2. Corrección en el eje Y "Altura"

EJE Y	
Distancia	**Corrección**
30	69
40	68
50	69
60	72
70	75
80	76
90	78
100	79
110	80
120	82
130	82
140	82
150	83
160	83
170	84
180	84
190	85
200	85
210	85
220	86
230	86
240	86
250	87
260	85
270	85
280	86
290	86
300	86
310	85
320	85
330	85

340	86
350	86
360	86
370	86
380	86
390	86
400	86

ANEXO C

C. PLANO ESTRUCTURA ROBOT MÓVIL

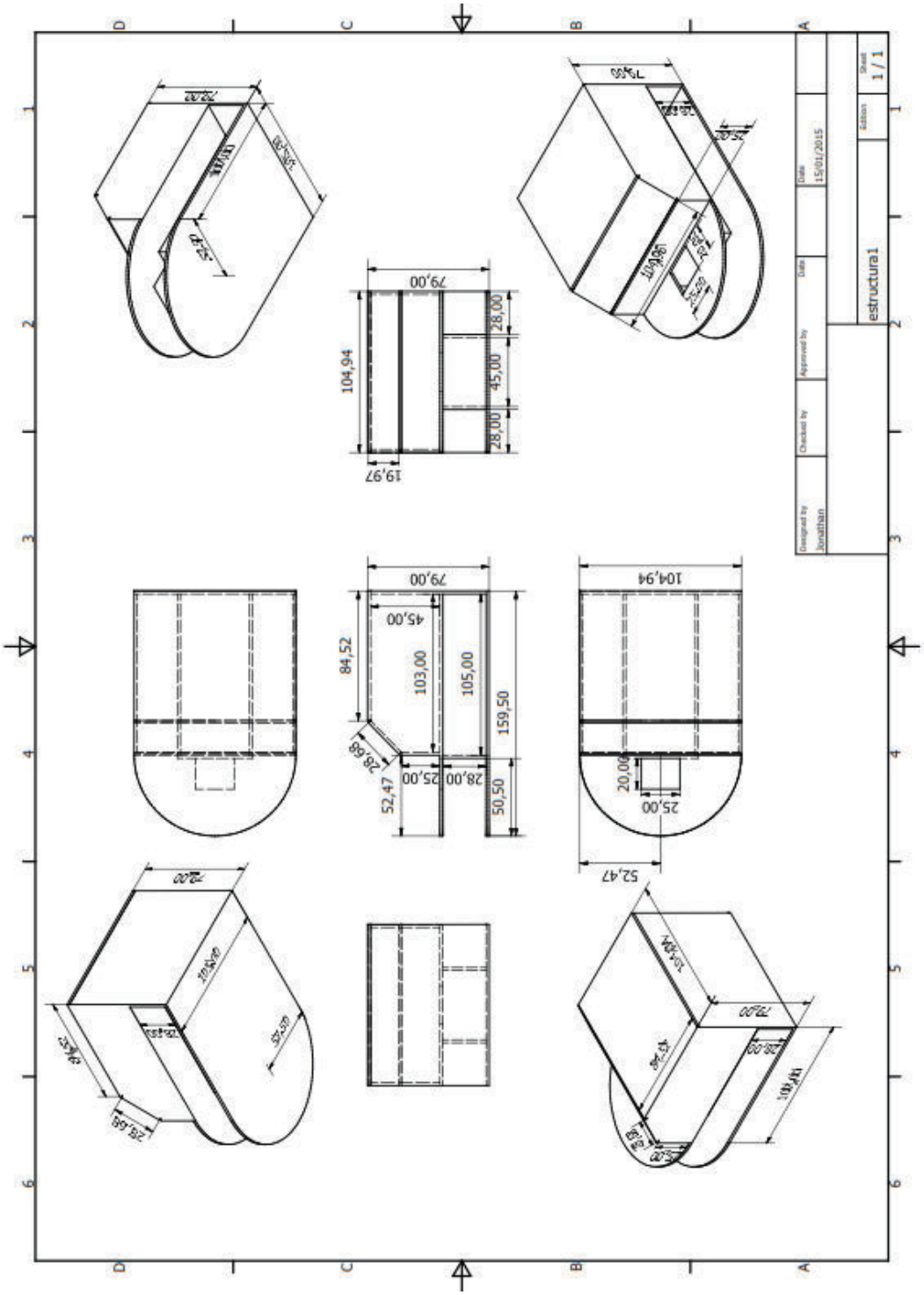


Figura C.1 Vistas de la estructura del robot en Autocad