UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:

INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:

INGENIERO ELECTRÓNICO

TEMA:

CONTROL DE UN ROBOT SEGUIDOR DE LÍNEA TIPO DIFERENCIAL MEDIANTE VISIÓN ARTIFICIAL

AUTOR:

JORGE LUIS CHANCO ALVEAR

TUTOR:

ANÍBAL ROBERTO PÉREZ CHECA

Quito, agosto del 2018

CESIÓN DE DERECHOS DE AUTOR

Yo, Jorge Luis Chanco Alvear, con documento de identificación N° 1724736481, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación intitulado: "CONTROL DE UN ROBOT SEGUIDOR DE LÍNEA TIPO DIFERENCIAL MEDIANTE VISIÓN ARTIFICIAL", mismo que ha sido desarrollado para optar por el título de Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

_____

Nombre: Jorge Luis Chanco Alvear

C.I. 1724736481

Fecha: Quito, agosto 2018

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de titulación "CONTROL DE UN ROBOT SEGUIDOR DE LÍNEA TIPO DIFERENCIAL MEDIANTE VISIÓN ARTIFICIAL" realizado por Jorge Luis Chanco Alvear, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerados como trabajo final de titulación.

Quito, agosto 2018

Aníbal Roberto Pérez Checa
C.I. 1711423440

DEDICATORIA

Dedico este trabajo en especial a mi querida hija, amada esposa y a quienes han estado siempre a mi lado en todo momento, brindándome su apoyo incondicional, también en especial a mi padre y madre quienes con su esfuerzo han hecho posible mi formación académica.

AGRADECIMIENTO

Agradezco a la Universidad Politécnica Salesiana por haberme acogido durante mis estudios de pregrado, así también a todos los profesores que estuvieron a mi lado, ya que con su experiencia y conocimientos han aportado en mi formación académica, un agradecimiento especial para mis padres quienes con su esfuerzo han permitido establecerme como profesional y como ente productivo para la sociedad, por otro lado, quiero agradecer a Luis y Rubén por todo el esfuerzo dado y su ayuda permanente en cada momento, también sin dejar de lado quiero agradecer a todas aquellas personas que en el diario vivir me supieron guiar y apoyar en los momentos difíciles de mi preparación.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Con el impulso de las técnicas de visión artificial aplicado al control de prototipos robóticos nace la necesidad de comprender, aprender, ensayar y desarrollar algoritmos que permitan el control de la navegación de robots seguidores de línea. En el presente trabajo se opta por la implementación de un prototipo de robot de pruebas para el ensayo y validación de algoritmos de visión artificial para los estudiantes del club de robótica de la Universidad Politécnica Salesiana sede Quito campus sur.

Basándose en estudios de los diferentes paquetes de software disponibles para el desarrollo de aplicaciones de visión artificial y mediante un análisis de las características técnicas de los diferentes sistemas embebidos, se determinó que la tarjeta Raspberry Pi es idónea para albergar los algoritmos de visión artificial aplicados al caso de estudio.

Por medio de la implementación del prototipo robótico junto con el algoritmo de visión artificial, se evaluó su efectividad por medio de pruebas de campo, para lo cual se diseñaron pistas con diferentes grados de dificultad, en las cuales dio como resultados una efectividad del 90% con trayectorias que posean curvas mayores a 10cm de radio y ángulos mayores a 90 grados.

Palabras claves: robot, prototipo, seguidor de línea, diferencial, visión artificial, sistemas embebidos, algoritmo.

ABSTRACT

With the impulse of the artificial vision techniques applied to the control of robotic prototypes, the need arose to understand, learn, test and develop algorithms that allow control of the navigation of line-following robots. In the present work we opted for the implementation of a test robot prototype, for the testing and validation of artificial vision algorithms for the students of the robotics club of the Salesian Polytechnic University headquarters Quito campus south.

Based on studies of the different software packages available for the development of artificial vision applications and by analyzing the technical characteristics of the different embedded systems, it was determined that the Raspberry Pi card is suitable to house the artificial vision algorithms applied to the case study.

Through the implementation of the robotic prototype together with the artificial vision algorithm, its effectiveness was evaluated by means of field tests, for which tracks with different degrees of difficulty were designed, in which it resulted in an effectiveness of 90% with trajectories that have curves greater than 10 cm radius and angles greater than 90 degrees.

Keywords: robot, prototype, line follower, differential, artificial vision, embedded systems, algorithm.

INTRODUCCIÓN

En el presente proyecto se implementó algoritmo de visión artificial basado en software libre, tal como lo es OpenCV para el seguimiento de líneas por parte de un robot móvil diferencial, el prototipo robótico adquiere imágenes por medio de una cámara la cual envía datos a una tarjeta de desarrollo Raspberry pi, la misma que alberga librerías y funciones para el procesamiento de la visión artificial, además realiza una comunicación con una tarjeta Arduino, la cual hace el control sobre un driver de motores para la locomoción del robot, en las pruebas realizadas al robot se verifico la eficiencia del robot y de los algoritmos implementados, logrando completar el 90% de las pruebas.

El planteamiento del problema, justificación y los objetivos que se han planteado en el presente proyecto, se encuentran en el Capítulo 1.

En el Capítulo 2, se describe se expone la robótica, en especial al robot seguidor de línea con tracción diferencial y su modelo matemático, se enfoca en la visión artificial y sus funciones para el procesamiento de imágenes, además se da una breve introducción a los sistemas embebidos.

En el Capítulo 3 se expone las tarjetas de desarrollo, características y la elección de la tarjeta compatible con sistemas de visión artificial y que se ajuste a las necesidades del proyecto se discrimino mediante tablas de ruta, además se presenta las características de los elementos tanto electrónicos como mecánicos que constituyen al robot, se expone el diseño estructural, sus dimensiones, se presentan los cálculos para el dimensionamiento de motores, por otro lado, se describe los algoritmos usados tanto para el sistema de visión artificial como para el control de motores

Finalmente, en el Capítulo 4, se hace referencia a las conclusiones y recomendaciones que se determinaron en el presente proyecto de titulación, los cuales son resultado de la implementación y de las pruebas realizadas al robot en diferentes pistas.

CAPÍTULO 1
ANTECEDENTES

1.1 Planteamiento del problema

Se ha observado que regularmente para el estudio de la robótica se construyen diferentes prototipos de robots, entre los más comunes están los seguidores de línea, pero hay que considerar que este tipo de robots tienen una baja precisión provocada por los sensores infrarrojos, los cuales se ven afectados por la luz del medio ambiente, sin tomar en cuenta los limitados métodos empleados para el control de dichos robots.

El club de robótica de la carrera de ingeniería electrónica sede Quito, requiere mejorar sus participaciones, pero se ha observado que se sigue compitiendo con robots seguidores de línea tradicionales y con los mismos métodos básicos de control. Con estos antecedentes y tomando en consideración la falta de buenos resultados en concursos bajo dicha categoría, se hace necesaria la implementación de un prototipo robótico con otros tipos de algoritmos, además que permita ensayar nuevos métodos de navegación, con el fin de ayudar al club de robótica a desarrollar sus habilidades en el desarrollo de algoritmos y por ende mejorar las participaciones en concursos en la categoría seguidor de línea.

1.2 Justificación del proyecto

Para el control y manejo de robots seguidores de línea de tipo diferencial se hace por medio de sensores que entregan datos de navegación puntuales, tal es el caso de los sensores infrarrojos o de proximidad, los cuales mantienen la limitación para su control.

El club de robótica de la UPS-UIO mantiene el uso de seguidores de línea con dichos sensores y con algoritmos de control sencillos, por lo tanto, frente a esta realidad se expone la necesidad de diseñar e implementar un prototipo de robot que sirva para implementación y prueba de algoritmos modernos de control, como el uso de la visión artificial.

1.3 Objetivos

1.3.1 Objetivo general

Implementar un algoritmo de visión artificial para el control de navegación de un robot seguidor de línea.

1.3.2 Objetivos específicos

- Desarrollar un marco teórico para identificar los algoritmos de visión artificial aplicables al caso de estudio.

- Identificar una placa controladora que soporte el procesamiento de imágenes para que la misma albergue los algoritmos de visión artificial que controlen el movimiento del robot.

- Implementar un robot tipo diferencial que sirva como plataforma de validación de los algoritmos identificados.

- Realizar pruebas de validación del sistema para comprobar la efectividad del mismo en diferentes líneas.

CAPÍTULO 2
MARCO TEÓRICO

2.1 Introducción

En el presente capitulo se presenta las generalidades de la robótica respecto a un robot seguidor de línea, tales como su locomoción, tipo de tracción y análisis cinemático del mismo, además se realiza una presentación a la visión artificial, sus conceptos, aspectos principales y los procesos generales que se usan en la implementación de la visión artificial para el caso de estudio.

2.2 Robot autónomo

Un robot con autonomía tiene la capacidad de interactuar en el entorno y cumplir con tareas establecidas, mediante la utilización de sensores que le brinda información del ambiente, sin intervención de algún agente externo.

Los robots autónomos cuentan con ciertas características principales, por ejemplo, una fuente propia de energía, normalmente baterías, otra característica es la de trabajar en entornos desconocidos, moverse sin ayuda humana y ser capaces de obtener información del entorno.

2.3 Robot seguidor de línea

Se caracteriza por tener como objetivo seguir una trayectoria trazada por una línea blanca o negra la cual contrasta con el color de fondo del suelo, es decir si la línea a seguir es negra, su fondo deberá ser blanco y viceversa.

Este tipo de robots son de gran popularidad lo que hace que existan competencias donde se fomenta el desarrollo de la robótica, en las cuales se añade diferentes dificultades a la categoría, como son: Líneas discontinuas, bifurcaciones, cuadriculas, subidas-bajadas, entre otras.

2.3.1 Sistema de locomoción

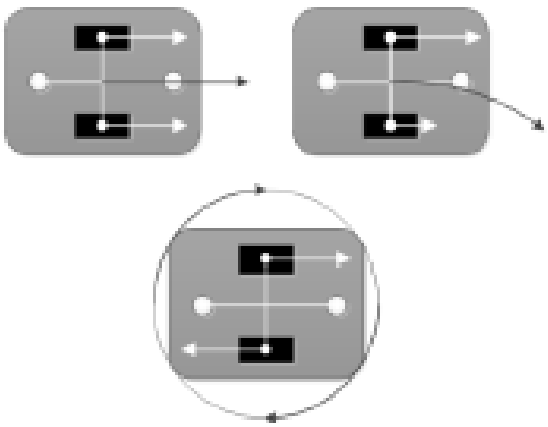
El sistema de locomoción es el modo en el que el robot se desplaza, el sistema más usado por los robots móviles es el de ruedas, que tiene ciertas ventajas como, por ejemplo: permite alcanzar mayores velocidades y menor costo. Por otro lado, cabe recalcar que el sistema de tracción diferencial es el más usado entre los robots

seguidores de línea por ser un sistema barato y de fácil implementación mecánica, además tiene como ventaja el realizar giros sobre su propio eje y por ende una mejor respuesta ante curvas cerradas.

2.3.1.1 Robot con tracción diferencial

El robot seguidor de línea con tracción diferencial se caracteriza por disponer de dos ruedas principales distribuidas paralelamente entre sí, cada rueda es independiente de la otra ya que mueve/gira cada una con un motor. Para desplazarse en línea recta ambas ruedas deben girar a la misma velocidad, y para realizar giros en este tipo de tracción, las ruedas deben tener una velocidad diferente una de la otra. Por otro lado, se dificulta estabilizar una plataforma o chasis con dos ruedas, por lo tanto, se coloca una rueda fija en el centro del lado opuesto de las ruedas principales.

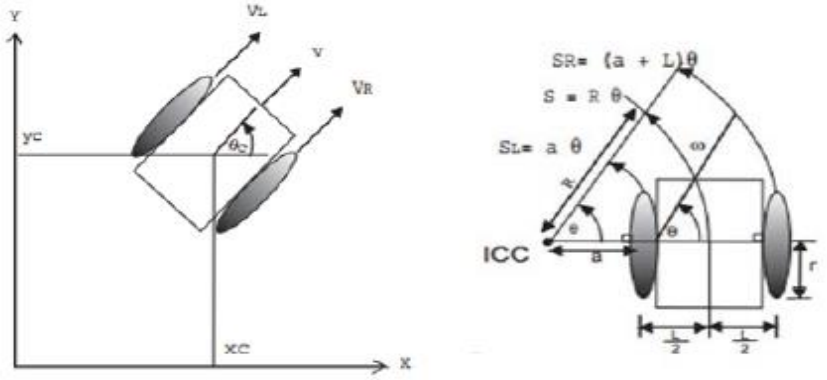
Figura 1.1. Configuración diferencial de un robot



Conducción y rotación de la configuración diferencial, Fuente: (Romo Caicedo & Toca Trujillo, 2016)

Para determinar el comportamiento del robot se parte del modelo cinemático del mismo, en la Figura 1.2 se muestran las variables involucradas con el movimiento del robot.

Figura 1.2. Modelo cinemático del robot móvil



Modelo cinemático de la tracción diferencial, Fuente: (Cárdenas Calle & Simbaña Quinde, 2007)

- θ_c: Ángulo del robot respecto a las coordenadas absolutas
- *Icc*: Centro instantáneo de curvatura
- ω: Velocidad angular del robot con respecto a su centro

Ecuación general de la velocidad en una rueda:

$velocidad = (radio * \omega)$ Ec. (1.1)

De la ecuación anterior se calcula las velocidades de las ruedas derecha e izquierda:

$V_{Izq.} = r * \omega L \;\; y \;\; V_{Der.} = r * \omega R$ Ec. (1.2)

Se calcula la distancia recorrida por la rueda izquierda, por la rueda derecha y la distancia recorrida por el centro del robot:

$S_L = a * \theta$: Distancia recorrida por la rueda izquierda Ec. (1.3)

$S_R = (a + L) * \theta$: Distancia recorrida por la rueda derecha Ec. (1.4)

$S = (R * \theta)$: Distancia recorrida por el centro del robot Ec. (1.5)

R es el radio de curvatura instantáneo del robot.

La Ec. (1.3) se reemplaza en la ecuación Ec. (1.4):

$S_R = S_L + L * \theta$ Ec. (1.6)

Para obtener ω se deriva la Ec. (1.6):

$$V_R = V_L + L * \omega \quad \text{Ec. (1.7)}$$

Se despeja ω de la ecuación anterior:

$$\omega = \frac{V_R - V_L}{L} \quad \text{Ec. (1.8)}$$

Se suma las ecuaciones Ec. (1.3) + Ec. (1.4) y se obtiene:

$S_L + S_R = a * \theta + a*\theta + L*\theta = 2*a*\theta + L*\theta$

$$S_L + S_R = 2 * \theta * \left(a + \frac{L}{2}\right) \quad \text{Ec. (1.9)}$$

De la Figura 1.2 , el radio de la curva:

$$R = \left(a + \frac{L}{2}\right) \quad \text{Ec. (1.10)}$$

Ahora al reemplazar Ec. (1.5) en la Ec. (1.9) se tiene:

$S_L + S_R = 2 * \theta * R \rightarrow S_L + S_R = 2 * S$

$$S = \frac{S_L - S_R}{2} \quad \text{Ec. (1.11)}$$

Derivando la ecuación anterior:

$$v = \frac{V_L + V_R}{2} \quad \text{Ec. (1.12)}$$

Calculo de R:

Derivando la Ec. (1.5)

$$v = \omega * R \quad \text{Ec. (1.13)}$$

Ahora se reemplaza las Ec. (1.12) y Ec. (1.8) en la anterior:

$$v = \frac{V_L + V_R}{2} = \frac{V_R - V_L}{2} * R$$

$$R = \frac{L}{2} * \frac{V_L + V_R}{V_R - V_L} \quad \text{Ec. (1.14)}$$

Analizando la ecuación, cuando las velocidades de las ruedas izquierda y derecha son iguales, R tiende al infinito, es decir que el robot va en línea recta. Si VR=−VL el radio es cero por lo que el robot girará alrededor de su centro.

A partir de las Ec. (1.12) y Ec. (1.8):

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \frac{V_L + V_R}{2} \\ \frac{V_R - V_L}{L} \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} V_L \\ V_R \end{bmatrix} \qquad \text{Ec. (1.15)}$$

Al multiplicar el Jacobino J(θ) con S se obtiene:

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta}^c \end{bmatrix} = \begin{bmatrix} \cos(\theta_c) & 0 \\ \text{sen}(\theta_c) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \text{J}(\theta_c)\text{s} \qquad \text{Ec. (1.16)}$$

Para obtener la posición se debe integrar la Ec. (1.16):

$$X_c = \int_0^t v(t)\cos(\theta_c)dT \qquad \text{Ec. (1.17)}$$

$$Y_c = \int_0^t v(t)\text{sen}(\theta_c)dT \qquad \text{Ec. (1.18)}$$

$$\theta_c = \int_0^t \omega(t)dT \qquad \text{Ec. (1.19)}$$

Y el centro Instantáneo de curvatura (ICC) está localizado en:

$$I_{cc} = ((X_c - R * sen(\theta_c),\ Y_c + R * cos(\theta_c)) \qquad \text{Ec. (1.20)}$$

2.4 Visión artificial

La visión artificial se define como un campo de la inteligencia artificial, la cual mediante técnicas permite la obtención, procesamiento y análisis de imágenes con el fin de obtener información. Con la visión artificial se pueden realizar diferentes actividades como la automatización de tareas repetitivas, control de calidad, inspecciones de objetos sin contacto físico, entre otras. (Gobierno de España, Ministerio de educacion, 2012)

Las funciones de la visión computacional en la robótica es dar seguridad, flexibilidad, verificación del sistema en general. Otro objetivo es permitir una adaptación del sistema cuándo existan cambios y aumentar la tolerancia a errores. (García Villanueva, Ramírez Zavala, & Ortega Reyes, 2016)

Los sistemas de visión artificial son un conjunto de procesos como: adquisición de imágenes digitales, trasformación y filtrado de imágenes, segmentación de objetos, etapa de reconocimiento e interpretación.
Las principales características de la visión artificial son:

- Medición del nivel de luz
- Detección de bordes y formas
- Análisis de color
- Se puede analizar un objeto en movimiento
- Sistemas basados en software

2.5 Software para visión artificial

Para el uso de visión artificial existen paquetes no comerciales, es decir software libre, entre los cuales destacan OpenCV, TargetJR, VXL, CVIP Tools, ImageLib, ImLib3D distribución de Linux, LookingGlass, NeatVision distribución de Java, TINA y XMegaWave distribución de Unix/Linux, Gandalf. Todos ellos disponen de herramientas para el procesamiento de imágenes, pero solo excepción de OpenCV y Gandalf proporcional un sistema completo para el desarrollo de aplicaciones con la Vision por computador.

Se puede decir que solo OpenCV provee de bibliotecas, herramientas para trabajar con la mayoría de las cámaras existentes en el mercado, además de entornos de desarrollo fáciles e intuitivos, ejecutándose en los sistemas más utilizados como son Windows y Linux. (Arévalo, González , & Ambrosio, 2002)

2.6 Requerimientos de hardware de OpenCV

OpenCV funciona en una amplia variedad de plataformas de hardware de la familia de Intel, tales como Atom, Xeon, entre otros. Los desarrolladores pueden programar OpenCV usando C++, C, Python y Java, además funciona en varios sistemas operativos como Windows, Linux, iOS y Android, hay que recalcar que algunas cámaras funcionan mejor que otras, esto depende de la compactibilidad del driver de la cámara con el sistema operativo. (Vinay, 2017)

Por otro lado, OpenCV se puede ejecutar en sistemas embebidos de tipo Linux, todo depende de la necesidad de procesamiento de la aplicación y de la complejidad del algoritmo ya que existen aplicaciones de bajo procesamiento y otras que necesitan gran cantidad de computo como por ejemplo Vision artificial en 4K y aplicaciones avanzadas.

2.7 OpenCV y visión artificial

OpenCV (Open Source Computer Vision Library) fue creado para el uso libre, tanto para fines comerciales como para fines investigativos, además tiene un enfoque de diseño para ser eficiente con el fin de tener aplicaciones en tiempo real, para OpenCV existen gran cantidad de librerías y más de 500 funciones de fácil utilización que potencian y ayudan a crear aplicaciones de visión artificial sofisticadas en diferentes áreas como interfaces hombre máquina, robótica móvil, seguridad-monitoreo, entre otras. (Bradski & Kaehler, 2008)

Para realizar visión artificial aplicada a la robótica móvil, en este caso un seguidor de línea es necesario el uso de ciertos métodos y procesos los cuales se nombran en este capítulo.

2.8 Preprocesado

El preprocesado sirve para reparar los errores producidos al momento de la captura de la imagen por parte de la cámara, ya sea por ruido, contraste o brillo, los algoritmos de preprocesado permiten corregir dichos errores mediante operaciones básicas, filtrado, entre otras. (Gonzáles Marcos, y otros, 2006)

Para la compresión del preprocesado y posteriores se utiliza como ejemplo una imagen de dos dimensiones de M*N y de 256 niveles de grises por punto, tomando en cuenta el valor 0 como negro carbón y el 255 como blanco.

Cada punto de la imagen se especificará por su posición i y j respecto a la esquina superior izquierda. Es decir: IM (i, j).

$$IM(i,j) \qquad \text{Ec. (2.21)}$$

2.8.1 Operaciones básicas del preprocesado

2.8.1.1 Clipping

Convierte los valores de grises superiores de su valor máximo val_{max} a un valor fijo (k_{max}), y los valores menores val_{min} a un valor mínimo k_{min}. Se resume de la siguiente forma en la Ec. (2.2).

$$IMB(i,j) = \begin{cases} K_{min} \ \ si \ \ IMA\,(i,j) < \ val_{min} \\ IMA(i,j) \ \ si \ \ val_{min} \leq IMA\,(i,j) \leq val_{max} \\ K_{max} \ \ si \ \ IMA\,(i,j) > \ val_{max} \end{cases} \qquad \text{Ec. (2.22)}$$

La función de Clipping permite realizar una igualación de todos los puntos que se consideran de un mismo origen, por ejemplo, en un fondo oscuro que tengan niveles de intensidad entre 0 y 20, es conveniente convertirlos todos a cero para simplificar su análisis. (Gonzáles Marcos, y otros, 2006)

2.8.1.2 Binarización e Inversión

Consiste dejar a cero todos los pixeles menores de un umbral, y poner en valor uno a los que son iguales o mayores, quedando de esta manera la imagen final en un conjunto de unos y ceros. La ecuación para implementar es:

$$IMB(i,j) = (IMA(i,j) \geq umbral) \qquad \text{Ec. (2.23)}$$

La inversión consiste en obtener el valor inverso (1/valor) del de cada pixel:

$$IMB(i,j) = 1/IMA(i,j) \qquad \text{Ec. (2.24)}$$

2.8.2 Segmentación

La segmentación es la división de imágenes en zonas disjuntas e individuales, es decir consiste en diferenciar diversos objetos, con el fin de comprender el contenido de la imagen. Hay dos formas principales de abordar la segmentación de imágenes: (Kenneth Dawson, 2014)

- Procesamiento de bordes, donde se identificó las discontinuidades en las imágenes.
- Procesamiento de región, donde se buscó regiones o secciones homogéneas de las imágenes.

2.8.2.1 Umbralización

Conocido como thresholding, es uno de los métodos más simples y eficientes de segmentación, lo cual significa que extrae o separa regiones de una imagen que sean de interés para el estudio, para ello se eliminan los valores superiores o inferiores respecto a un valor conocido como umbral según el caso. La expresión matemática se puede observar en la siguiente ecuación. (Gonzáles Marcos, y otros, 2006)

$$IMB(i,j) = IMA(i,j)) * (IMA(i,j) \geq umbral) \quad \text{Ec. (2.25)}$$

La Umbralización adaptativa permite establecer un valor de umbral que se calcula o varía según las características del entorno a evaluar, esto permite segmentar imágenes que contengan fondos con distintos niveles de grises o iluminación no uniforme.

2.8.3 Detector de bordes Canny

El algoritmo Canny tiene como objetivo cumplir principalmente con los siguientes criterios: (Huamán , 2018)

- Baja tasa de errores: una buena detección de solo los bordes existentes.
- Buena localización: Minimización de bordes detectados y los bordes reales
- Respuesta mínima: solo una respuesta del detector por borde.

El algoritmo Canny es de múltiple etapa, las cuales son:

- Reducción de ruido: mediante un filtro gaussiano de tamaño 5x5.
- Encontrar la gradiente de intensidad de la imagen: que mediante el kernel Sobel se filtra en dirección horizontal y vertical, para conseguir la derivada en ambas direcciones, a partir de ello se puede encontrar el gradiente borde y la dirección de cada pixel.

$$Borde\ gradiente\ (G) = \sqrt{G_x^2 + G_y^2}$$ Ec. (2.26)

$$Ángulo\ (\theta) = tan^{-1}(\frac{G_y}{G_x})$$ Ec. (2.27)

- Supresión no máxima: se escanea la imagen y se elimina los pixeles no deseados que pueden no constituir el borde.

- Umbral de histéresis: en esta etapa se elige los bordes que realmente son bordes y cuáles no, entonces los bordes con un valor de intensidad mayor que el valor de umbral seguramente serán bordes.

2.8.4 Contornos

Aunque los algoritmos como el detector de bordes de Canny se puede usar para encontrar los píxeles de los bordes que separan diferentes segmentos en una imagen, no da ninguna información sobre esos pixeles como entidades en sí mismos.

Por lo tanto un contorno es un conjunto de puntos que conectado forman una figura que rodea un objeto determinado, la detección de contornos en OpenCV se aplica sobre imágenes binarias, la figura resultante que se obtiene puede ser analizada para determinar el objeto detectado, para OpenCV existe la función cvFindContours () para ensamblar esos pixeles de borde en contornos. (Bradski & Kaehler, 2008)

2.8.5 Secuencias

Un tipo de objeto que se puede almacenar dentro de un almacenamiento de memoria es una secuencia. Las secuencias son listas vinculadas de otras estructuras. OpenCV puede hacer secuencias de muchos tipos diferentes de objetos.

2.8.5.1 Contornos como secuencias

Cuando se llama a la función de *OpenCV: cvFindContours ()*, nos dará una gran cantidad de secuencias las cuales son puntos; más precisamente, son contornos. En particular, son secuencias de puntos que representan algún tipo de curva en el espacio de la imagen. (Kenneth Dawson, 2014)

2.8.5.2 Momentos

Una de las formas de comparar dos contornos es calcular los momentos del contorno. En términos generales, un momento es una característica bruta del contorno, sumando todos los píxeles del contorno. (Bradski & Kaehler, 2008)

El momento en una imagen con intensidades de pixeles $I(x, y)$, los momentos de una imagen pura M_{ij} son calculados por:

$$M_{ij} = \Sigma_x \Sigma_y I(x, y) \qquad \text{Ec. (2.28)}$$

Donde i, j son orden de momentos. En una imagen típica el cálculo de momento de, la imagen $I(x, y)$ se convierte en un conjunto de vectores, calculando primero una imagen en escala de grises y luego usando un detector de esquinas o borde para reducir la imagen a un conjunto de puntos sobresalientes. Los diversos ordenes de momentos de la imagen, es decir el conjunto de puntos salientes, proporcionan masa, centroide, eje principal, y demás de la imagen.

2.9 Sistemas embebidos

Un sistema embebido es un sistema electrónico diseñado para realizar pocas funciones en tiempo real, al contrario de las computadoras que cubren un amplio rango de necesidades y operaciones los sistemas embebidos se diseñan para cumplir funciones específicas. Dichos sistemas no solo están desplegados en la industria, sino que existen en varios equipos de uso diario como automóviles, ascensores, sistemas de seguridad entre otros.

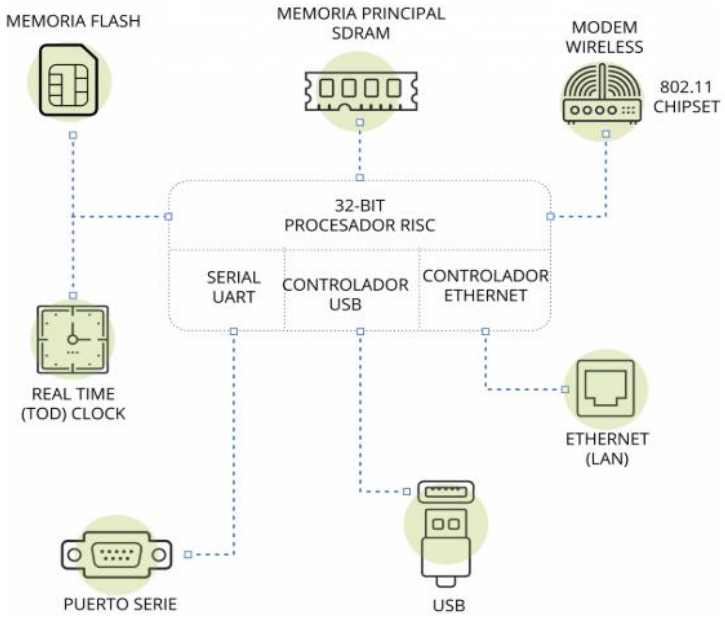
Los sistemas embebidos están constituidos de tal manera, que se pueden programar directamente en el lenguaje de microcontrolador o microprocesador incorporado sobre el mismo, también se usan compiladores específicos que utilizan lenguajes como C o C++. El sistema embebido está conformado por un microprocesador además de memorias RAM o ROM donde se ejecuta y guarda un software que usa el sistema embebido. (Cruz, 2013)

2.9.1 Características de los Sistemas Embebidos

Las principales características de un sistema embebido es su bajo consumo energético y también su bajo costo, dado que este tipo de sistemas son diseñados para

ser producidos en masa. Hay que recalcar que estos sistemas tienen dos grandes funcionalidades como la fiabilidad, ya que son sistemas capaces de funcionar de manera ininterrumpida, y la personalización, ya que existe la posibilidad de modificar el sistema de acuerdo con las necesidades del proceso. (INCIBE, 2018) En la Figura 2.3 se observa un sistema embebido a nivel físico.

Figura 2.3. Descripción del sistema embebido



Partes físicas de un sistema embebido, Fuente: (INCIBE, 2018)

Las ventajas de los sistemas embebidos se nombran a continuación:

- Utilización de sistemas operativos potentes tales como: Linux, Windows los cuales pueden realizar numerosas tareas tales como: comunicación por redes IP, soporte grafico entre otras.

- Al utilizar sistemas embebidos se pueden encontrar fácilmente un gran número de herramientas de desarrollo, así como una gran comunidad de programadores que dominan dichos sistemas, dada la extensión mundial de las aplicaciones compatibles para computadoras.

- Reducción de precio en componentes de hardware y software debido a la construcción en masa y gran cantidad de computadoras en el mundo.

CAPÍTULO 3
DISEÑO DE HARDWARE DEL ROBOT

El objetivo principal de este trabajo es el control del seguidor de línea mediante la implementación de algoritmos de visión artificial, y para ello se requiere del uso de procesamiento de imágenes. Además, es necesario un sistema de tipo portable que pueda ser albergado en un robot, que sea capaz de alojar librerías de OpenCV; También se requiriere la comunicación con periféricos como: una cámara, mini teclado-mouse y las conexiones a otros dispositivos o drivers, que permitan concebir el control para el movimiento del robot.

3.1 Selección de las tarjetas de desarrollo

Para la selección de la placa controladora se escogió entre ciertas tarjetas de desarrollo, las cuales se exponen a continuación. La selección de dicha placa se realizó basándose principalmente en ciertos criterios de hardware que sirvan para cumplir con la implementación del proyecto; Además, el costo y la disponibilidad en el mercado ecuatoriano, que sea compatible con software de visión artificial, interfaz de comunicación con otras tarjetas y periféricos externos, de preferencia que se compatible con sistemas operativos de tipo código abierto, con el fin realizar cambios o mejoras al proyecto en trabajos futuros.

3.1.1.1 BeagleBone Black

La BeagleBone Black, ver Figura 3.1, es una tarjeta de desarrollo con un CPU ARM de 1GHz y 512 MB en RAM, conectividad Wireless, con sistema de base Linux, lo que realmente destaca a esta tarjeta es sus capacidades de expansión con interfaces tipo Shields, el costo aproximado es 130 USD en los Estados Unidos, con poco mercado en Ecuador. (Griffith, 2017)

Figura 3.1. BeagleBone Black

BeagleBone Black Rev. C, Fuente: (Griffith, 2017)

3.1.1.2 Raspberry Pi 3

La tarjeta de desarrollo Raspberry ver Figura 3.2, cuenta con grandes capacidades un CPU ARM x64 Cortex de 1.2GHz y 1GB RAM, con sistema Linux, conectividad Wireless y bluetooth, cuenta con capacidad de conexión con dispositivos externos mediante USB y con un costo accesible de 80 USD disponible en el mercado ecuatoriano. (Griffith, 2017)

Figura 3.2. Raspberry Pi 3

Raspberry Pi 3 Modelo B, Fuente: (Griffith, 2017)

3.1.1.3 Joule 570X de Intel

La Joule 570X, ver Figura 3.3, cuenta con un CPU Atom T5700 de cuatro núcleos con un reloj base de 1,7 GHz, se puede expandir a 4 GB de RAM. Admite la reproducción y captura de video en 4K, si bien la placa Joules ofrece un gran rendimiento para cualquier diseño, fue creada para ocupar un lugar específico en el mercado de las placas de desarrollo ya que según sus fabricantes está dirigida a desarrolladores avanzados de aplicaciones de visión artificial y video de mayor rendimiento en base Linux, tiene un costo aproximado de 584 USD en los Estados Unidos con una inexistente demanda en el mercado ecuatoriano. (Griffith, 2017)

Figura 3.3. Tarjeta Joule 570X

Joule 570X de Intel, Fuente: (Griffith, 2017)

3.1.1.4 Tarjeta DECA

La placa DECA ver Figura 3.4 , es una FPGA, posee un solo chip con un procesador Nios II Gen2 de 32 bits RISC de Altera, 512MB de RAM, además de conectividad USB y ethernet, cabe recalcar que funciona con sistema Windows 7 y la tarjeta tiene un costo de 200 USD en los Estados Unidos, sin disponibilidad en el mercado Ecuatoriano. (Griffith, 2017)

Figura 3.4. Tarjeta DECA

FPGA DECA, Fuente: (Griffith, 2017)

3.1.1.5 Placa AM3359 de Texas Instruments

La AM3359, ver Figura 3.5, es una placa de desarrollo con un potente procesados ARM Cortex A8 Sitara AM335x, además tiene una unidad de tiempo real programable (PRU) integrada, cabe recalcar que la tarjeta trabaja con el sistema operativo Windows, la placa es especialmente diseñada por Texas Instruments para tecnologías de comunicación en tiempo real, a la tarjeta se le puede añadir una pantalla OLED integrada, ya que tiene un gran rendimiento

de energía, tiene un costo aproximado de 1053 USD, sin mercado en Ecuador. (Griffith, 2017)

Figura 3.5.Tarjeta AM3359

Placa AM3359, Fuente: (Griffith, 2017)

En la Tabla 3.1 se puede observar un resumen de las características relevantes de las tarjetas anteriormente expuestas.

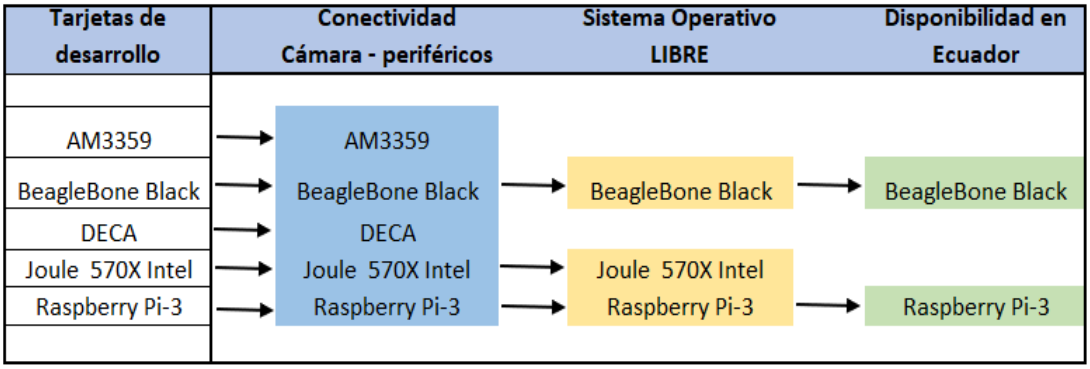
Tabla 3.1. Resumen de características de tarjetas

Tarjetas de desarrollo	Tipo de procesador y/o velocidad	Memoria RAM	Conectividad Cámara - periféricos	Sistema Operativo LIBRE	Precio aproximado en USA	Disponibilidad en Ecuador
AM3359	ARM Cortex	512 MB	✓	x	1053 USD	x
BeagleBone Black	ARM de 1.0 GHz	512 MB	✓	✓	130 USD	✓
DECA	FPGA	512 MB	✓	x	200 USD	x
Joule 570X Intel	Atom de 1.7 GHZ	1 GB	✓	x	584 USD	x
Raspberry Pi-3	ARM de 1.2 GHz	1 GB	✓	✓	80 USD	✓

Resumen de características, Elaborado por: Jorge Chanco

Para la elección de la tarjeta de desarrollo se realizó una tabla de ruta (fixture), para clasificar las tarjetas que cumplen con las necesidades del proyecto, ya que OpenCV se puede ejecutar en cualquier sistema embebido que ejecute un sistema operativo, todas las tarjetas nombradas anteriormente son capaces de ejecutar OpenCV, para el fixture se tomó en cuenta la comunicación con periféricos de entrada, un sistema operativo libre en base Linux para usar aplicaciones libres y sin limitación de licencias, también se basó en la disponibilidad en el mercado ecuatoriano. En la Figura 3.6 se observa la ruta clasificatoria de las tarjetas embebidas.

Figura 3.6. Fixture clasificatorio



Fixture clasificatorio de tarjetas embebidas, Elaborado por: Jorge Chanco

Se obtuvo que las tarjetas mejor cualificadas son las BeagleBone Black y Raspberry Pi3, pero para elegir entre las dos tarjetas se basó en las características de hardware que se observan en la Tabla 3.1,y se determinó que la Raspberry Pi3 tiene el doble de RAM que la tarjeta BeagleBone y por lo tanto tiene mayor velocidad, lo que ayuda al procesamiento de la visión artificial.

Basándose en dichos antecedentes y tomando en cuenta las necesidades del proyecto y su alcance, la tarjeta Raspberry Pi 3 es la adecuada para ejecutar OpenCV y por ende sirve para el reconocimiento y seguimiento de líneas, al ser una microcomputadora posee las capacidades para adquirir imágenes de la cámara, identificar y calcular la posición de la línea a seguir. Cabe recalcar que la tarjeta funciona con sistema Debian basado en Linux lo cual la hace compatible con las herramientas de programación para OpenCV.

La tarjeta Raspberry Pi 3 modelo B es la última versión de la organización Raspberry, y básicamente es una microcomputadora, en la cual se puede instalar una de las versiones del sistema operativo Linux, las versiones compatibles se pueden descargar directamente en la página web de la organización.

3.2 Diseño electrónico del robot

Para el reconocimiento de líneas se usará la herramienta Open CV implementado en la tarjeta Raspberry Pi3. Para el control del robot se lo separo en dos partes, la primera que es el algoritmo de visión artificial junto con la interfaz del usuario y la segunda parte la cual realizará el control de la locomoción del robot, mediante un driver, el cual es controlado por una tarjeta de desarrollo de menor gama y de uso específico. Por otro lado, también se busca un cierto nivel de autonomía por parte del robot ya que al ser un prototipo de pruebas es necesaria dicha característica, por lo cual se expone más adelante en este capítulo el sistema de alimentación, y el tiempo aproximado de autonomía.

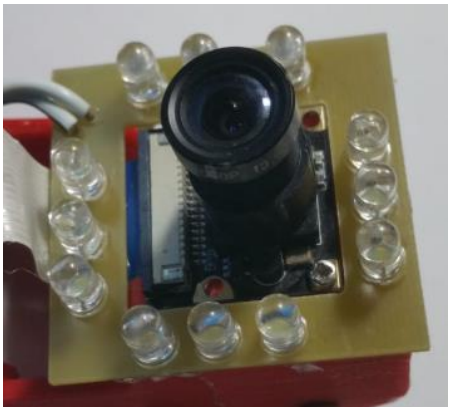
En resumen, el robot está conformado por varios sistemas electrónicos tales como: el sistema de control, driver de potencia para accionar los motores, sistema sensorial, sistema de interfaz con el usuario, sistema de alimentación.

3.2.1 Sistema sensorial

Para realizar la captura de imágenes de la línea a ser seguida, se necesita una cámara que sea compatible con la tarjeta Raspberry Pi3 y que permita obtener datos para ser analizados con la herramienta de OpenCV, con dichos antecedentes se eligió una cámara propia de Raspberry Pi3, que ofrece una conexión directa entre la tarjeta y la cámara mediante una interface CSI (Camera Serial Interface), además para evitar sombras no deseadas al momento de captura de imágenes que son utilizadas por la visión artificial de OpenCV, se montó un arreglo de leds en contorno a la cámara, ver Figura 3.7. El circuito y PCB está en el Anexo 1. Las especificaciones principales de la cámara se detallan a continuación:

- Totalmente compatible con Raspberry Pi
- 5 megapíxeles Omnivision
- Resolución de imágenes fijas :2592 x 1944
- Angulo de visión de 69.9 grados aproximadamente
- Se puede añadir infrarrojos de 3w de potencia
- Resistencia ajustable a bordo para controlar el Umbral de luz ambiental.
- Focal ajustable

Figura 3.7. Cámara focal 5MP



Cámara focal 5MP, Fuente: Jorge Chanco Alvear

3.2.2 Sistema de control

Como se mencionó anteriormente en este capítulo, se usará la Raspberry para el algoritmo de visión artificial, por lo tanto, se nombran las características específicas del microcomputador Raspberry Pi (Heath , 2017):

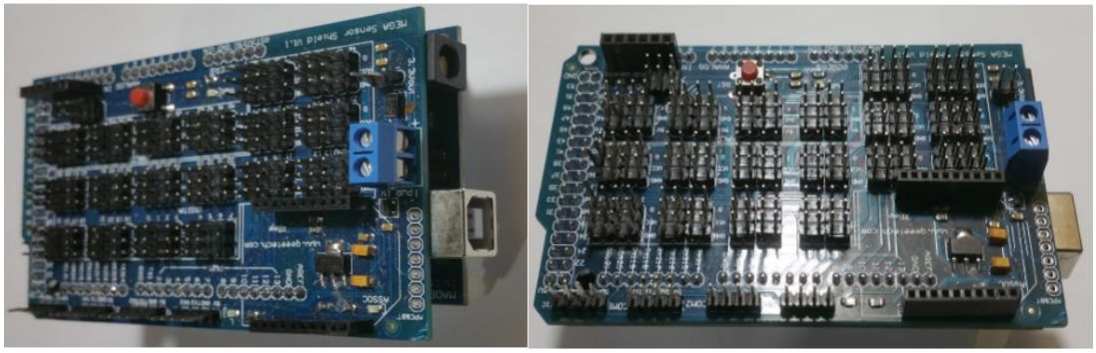
- Chipset: BCM2837
- CPU: 1.2GHz Quad Core 64-bits ARM Cortex
- Ethernet: 10/100
- USB: 4 USB 2.0.
- Wireless:802.11n Wireless LAN, Bluetooth 4.1
- Memoria: 1GB LPDDR2-900 SDRAM
- Expansible: 40 pines de propósito general
- Video: puerto HDMI
- Interfaz Cámara (CSI)
- Interfaz de display

Para el control del driver de los motores se optó por usar la plataforma libre Arduino Mega, con el fin de descentralizar el proceso de control y proteger a los puertos de entrada y salida de la raspberry de posibles retornos de corriente y del ruido electromagnético. El Arduino Mega posee un microcontrolador ATmega2560, con una conexión USB para ser fácilmente integrada a la Raspberry, además es compatible con shields de expansión. Las especificaciones técnicas más relevantes se nombran a continuación:

- Microcontrolador Atmega2560
- 39 pines de E/S digitales
- 15 pines para PWM
- 16 entradas analógicas
- Velocidad de reloj 16Mhz
- Corriente DC por pin E/S: 20 mA
- SRAM: 8 KB
- Voltaje de alimentación: 5.1v

Como parte del diseño electrónico se implementa el Arduino Mega Sensor Shield, cuyo fin es facilitar las conexiones entre el Arduino Mega y el driver de los motores, además facilitar futuras expansiones de hardware al robot, ya que este Shield permite conectar varios módulos o sensores simultáneamente al Arduino. En la Figura 3.8 se puede observar una fotografía del sensor Shield, junto al Arduino Mega.

Figura 3.8. Sensor Shield y Arduino Mega



Sensor Shield - Arduino Mega, Fuente: Jorge Chanco Alvear

3.2.3 Sistema Interfaz

Para que el usuario consiga interactuar directamente con el modo gráfico de la tarjeta Raspberry y pueda realizar ejecuciones en el sistema, se adiciono una pantalla a color de 5 pulgadas, es especialmente diseñada para Raspberry Pi 3, ya que se conecta directamente por medio de su puerto HDMI, además de una conexión por medio de comunicación SPI para habilitar el uso del touch panel que posee algunas pantallas, a continuación, se detalla sus características:

- Alimentación mediante GPIOs de Raspberry
- Tipo de pantalla: TFT LCD de 5 pulgadas
- Resolución:800x480
- Dimensiones: 120mm x 74mm x 7mm
- Conexión: HDMI y SPI

Tomar en cuenta que a la tarjeta se le puede conectar directamente periféricos de entrada y salida por medio de USB, tales como teclado y mouse, lo que facilita la navegación en el entorno grafico de Raspberry, en el presente proyecto se usó un mini teclado-mouse inalámbrico USB, para la interacción con la tarjeta. En la Figura 3.9 se puede observar el mini teclado-mouse inalámbrico y la pantalla de 5 pulgadas.

Figura 3.9. Pantalla de 5 pulgadas

Pantalla de 5 pulgadas de Raspberry pi, Fuente: Jorge Chanco Alvear

3.3 Aspectos para la selección del chasis del robot

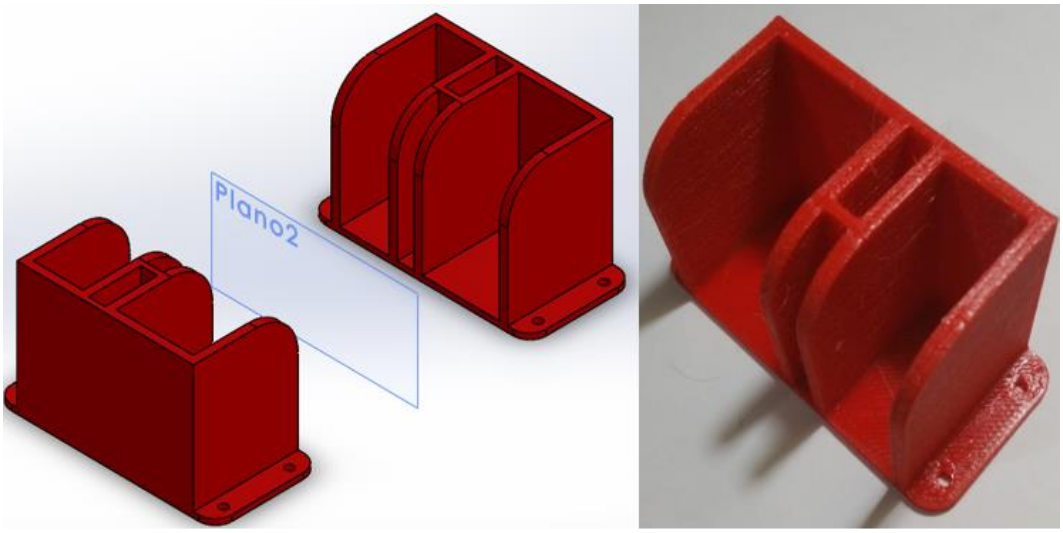
Tomando en cuenta las dimensiones de los elementos que conforman al robot, se implementó un chasis que pueda almacenar todos los elementos, además ya que es un prototipo de pruebas es deseable cierta dureza del chasis, por lo tanto la estructura es relativamente grande y pesada, se tomó en cuenta la autonomía deseable del robot, lo que implicó añadir baterías de alta capacidad relativamente grandes y pesadas, por lo tanto con dichos antecedentes, en el robot se estimó una masa aproximada 5kg , con lo cual se calculó el torque necesario de los motores para mover al robot y por ende la selección del driver de los mismo.

3.3.1 Diseño computarizado del chasis del robot

Como se mencionó anteriormente en el presente capítulo, el tamaño del chasis procede de la cantidad de elementos que posee el robot, para el diseño del chasis se ubicó de diferentes posiciones los elementos electrónicos hasta lograr una distribución práctica que permita obtener el tamaño del chasis final y que pueda almacenar todos los componentes, como se observa en la Figura 3.12, además en el diseño se optó por ubicar la pantalla de la Raspberry en la tapa superior de robot como se observa en la Figura 3.11 para que el usuario acceda a la interfaz gráfica de la tarjeta. Por otro lado, se consideró, que al ser Raspberry una microcomputadora, sus elementos electrónicos disipan calor, junto con los elementos de potencia del driver de motores, por lo tanto, es necesario la implementación de ventilación forzada para la circulación de aire dentro del robot, para ello en las tapas laterales del robot, tiene cortes para la entrada y salida de aire mediante ventiladores. Los planos del chasis se pueden observar en el Anexo 2.

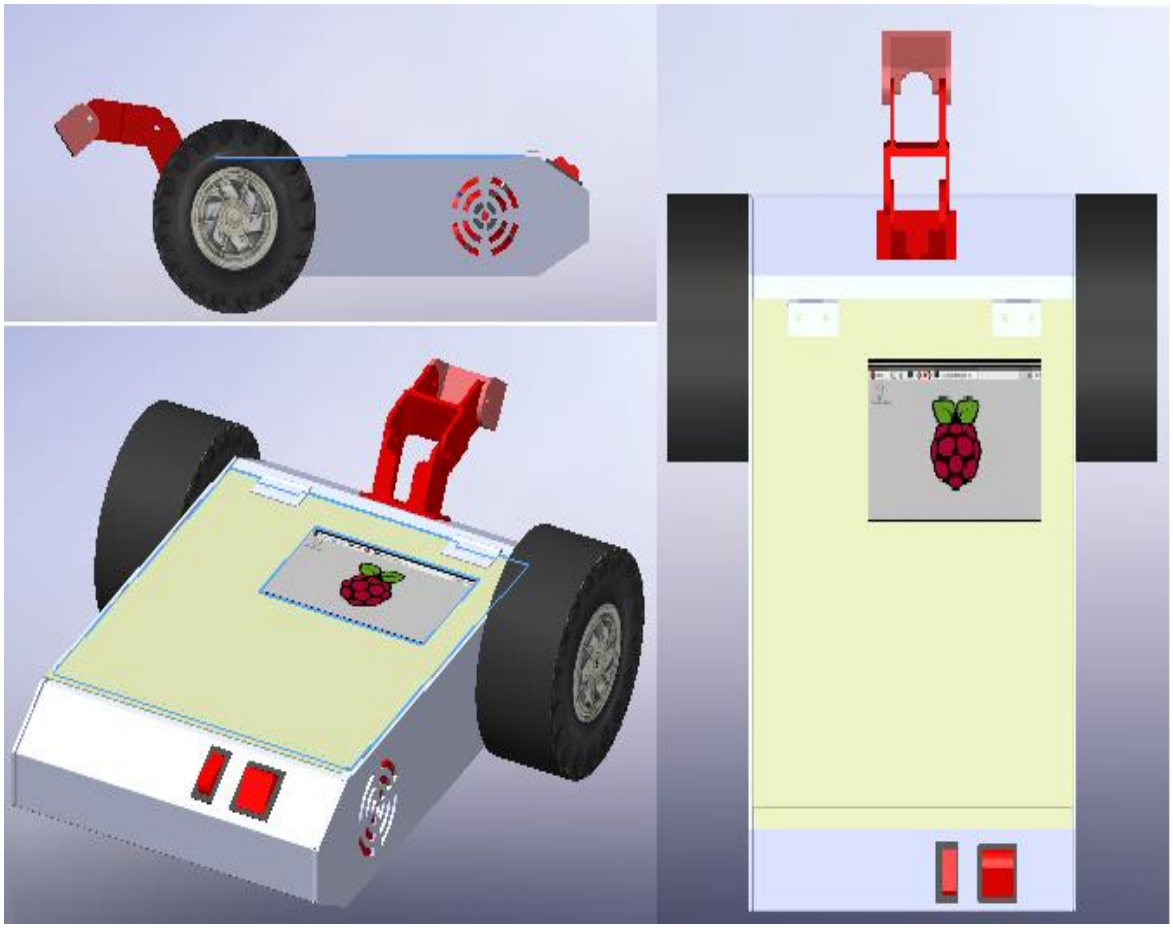
Para las baterías se consideró realizar soportes, los mismos que facilitan la ubicación y extracción de las mismas. En la Figura 3.10 se puede observar los soportes y los planos se pueden observar en el Anexo 3.

Figura 3.10. Soportes de las baterías del robot



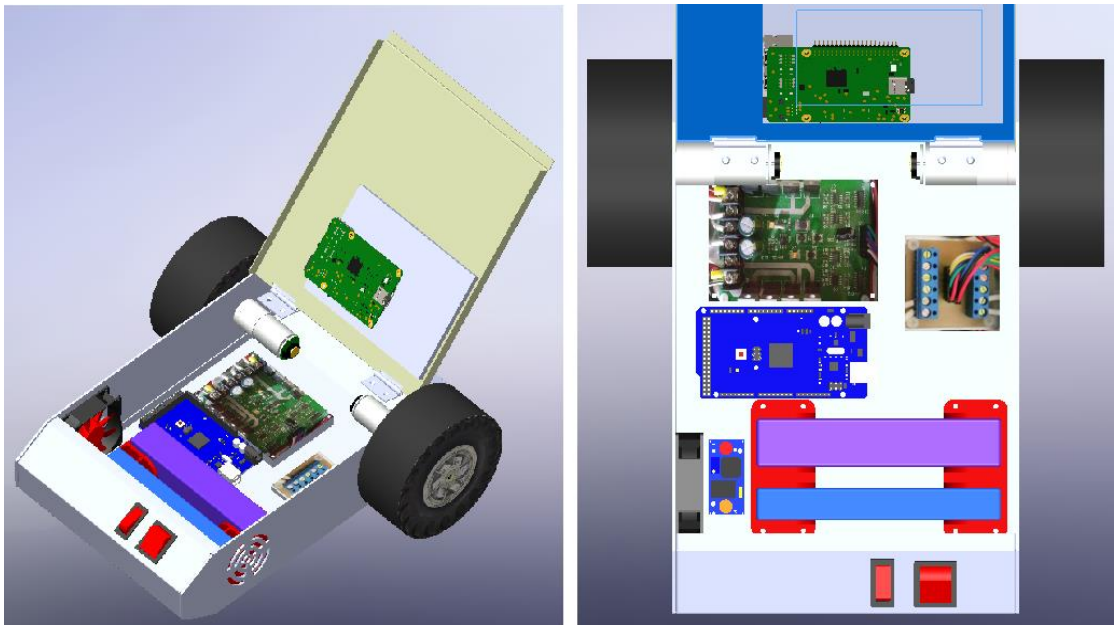
Soportes baterías diseñadas con SolidWorks, Fuente: Jorge Chanco Alvear

Figura 3.11. Chasis del robot



Diferentes vistas del robot en el entorno de SolidWorks, Elaborado por: Jorge Chanco Alvear

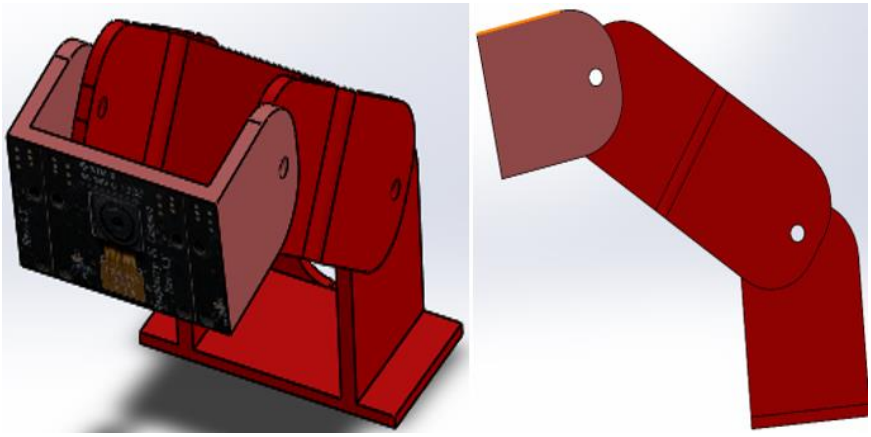
Figura 3.12. Vistas internas del Chasis del robot



Vistas de componentes del robot en el entorno de SolidWorks, Fuente: Jorge Chanco Alvear

Para la ubicación de la cámara Pi en el robot, y con el fin de encontrar el mejor ángulo para la captura de imágenes se optó por diseñar un soporte articulado para la cámara, el cual pueda variar su ángulo de inclinación. El soporte se lo observa en la Figura 3.13 y va ubicado en la parte superior frontal del robot como se muestra en la Figura 3.11.

Figura 3.13. Vistas Soporte cámara Pi



Soporte de la cámara Pi diseñadas con SolidWorks, Elaborado por: Jorge Chanco Alvear

3.4 Motores y ruedas robot

Para la selección de los motores es necesario calcular el torque requerido para conseguir el movimiento del robot, se empleó la Ec. (3.1) obtenida en (Neal, 2010), para ello, es necesario conocer ciertos datos, tales como la masa total del robot, el radio de la rueda y la aceleración deseada.

$$T = M * (a + g * \sin(\theta)) * r \qquad \text{Ec. (3.1)}$$

Donde:

- T: Torque del motor.
- T: Torque del motor.
- M: Masa total del robot.
- a: Aceleración.
- θ: Ángulo del plano.
- g: Gravedad.
- r: Radio de las ruedas.

Se asume una masa de 5 Kg aproximadamente, según las estimaciones de los pesos de los componentes, tanto estructuralmente como en componentes electrónicos y baterías de larga duración que lleva el robot, el radio de las ruedas es de aproximadamente 7 cm, y se desea una aceleración aproximada de 0.5 m/s^2, ya que es un prototipo de pruebas y de campo, el ángulo θ seria variante, pero para el caso de estudio será un ángulo de cero ya que se lo probará en una pista totalmente plana.

$$T = 5\ Kg * 0.5\frac{m}{s^2} * 0.07m = 0.175\ N * m$$

Al ser motores de marca Pololu, la tabla de las características de torque viene dada en oz/in, por lo tanto, se realiza una conversión de unidades.

$$T = 0.175N * m * \frac{0.225\ lb}{1N} * \frac{16\ oz}{1\ lb} * \frac{100cm}{1m} * \frac{1cm}{2.54\ in} = 24.80\ oz * in$$

$$\mathrm{T} = 24.80\ \mathrm{oz} * \mathrm{in}$$

Debido a la velocidad de procesamiento de imágenes es necesario que el robot recorra la trayectoria a una mínima velocidad, por lo tanto, como se observa en la Figura 3.14 el motor con la menor velocidad es de 100 RPM, con una relación en su caja de engranajes de 99:1 con una alimentación de 12v DC, cuyas características son suficientes para cubrir con las necesidades del robot.

Para calcular la velocidad máxima que podría alcanzar el robot, tomando en consideración una rueda de 7 cm de radio se calcula con la Ec. (3.2)

$$V = \frac{V_{RPM}}{60_{seg}} * 2\pi * r \qquad \text{Ec. (3.2)}$$

$$V = \frac{100}{60_{seg}} * 2\pi * 0.07_m$$

$$V = 8.333\ \frac{0.07_m}{seg}$$

$$V = 0.73\ \frac{m}{s}$$

El valor de la velocidad máxima calculada que podría alcanzar el robot es en condiciones ideales, es decir, un valor netamente teórico.

Figura 3.14. Características eléctricas y mecánicas micromotor Pololu 47:1

Rated Voltage	Motor Type	Stall Current @ Rated Voltage	No-Load Speed @ Rated Voltage	Approximate Stall Torque @ Rated Voltage	 With Encoder
12 V	high-power (HP)	5.6 A	10,200 RPM	5.5 oz-in	1:1 HP 12V w/encoder
			2250 RPM	23 oz-in	4.4:1 HP 12V w/encoder
			1030 RPM	44 oz-in	9.7:1 HP 12V w/encoder
			500 RPM	85 oz-in	20.4:1 HP 12V w/encoder
			290 RPM	120 oz-in	34:1 HP 12V w/encoder
			210 RPM	165 oz-in	47:1 HP 12V w/encoder
			130 RPM	240 oz-in	75:1 HP 12V w/encoder
			100 RPM	300 oz-in	99:1 HP 12V w/encoder

Catálogo de micromotores Pololu, Fuente: (Pololu Robotics and Electronics, 2018)

Como se mencionó anteriormente en esta sección el radio de las ruedas es de 7cm, incluyendo el rin y su recubrimiento, tomando en cuenta que el prototipo robótico va a ser para pruebas y ensayos de algoritmos, se eligió un tipo de ruedas todoterreno para atravesar terrenos complejos y accidentados, de tal manera que el robot pueda desenvolverse en ambientes de prueba, las ruedas tienen un tipo de neumático hecho de caucho suave y con puntas para una mayor tracción, además la rueda posee una fácil adaptación al eje de 4 mm de diámetro del motor ,en la Figura 3.15 , se puede observar la rueda de caucho seleccionada y el correspondiente motor.

Figura 3.15. Rueda tipo todoterreno

Rueda todoterreno Pololu, Fuente: (Pololu Robotics and Electronics, 2018)

3.4.1 Encoder

Para conocer el estado de los motores es necesario un encoder que viene acoplado al eje del micromotor, ver Figura 3.16, este sensor es de efecto Hall, y posee dos salidas A y B con un desface de 90 ° entre salidas, con señales de salida en cuadratura con una resolución de 48 pulsos por revolución del eje del motor. El sensor Hall requiere una tensión de entrada, Vcc, entre 3.5 y 20 V y consume un máximo de 10 mA.

Figura 3.16. Metal Gearmotor 47:1 con encoder de efecto Hall

Metal Gearmotor 47:1, Fuente: (Pololu Robotics and Electronics, 2018)

3.4.2 Controlador de potencia

Para el accionar de los motores es necesario un driver de potencia, el cual se encarga de suministrar el voltaje y corriente necesaria para los mismos, para la elección del driver se tomó en cuenta tanto la corriente de arranque como la nominal de los motores, en la Figura 3.14 se observa una corriente máxima del motor de 5.6A a 12v, este dato se toma en consideración para la elección del driver, se revisó algunas opciones disponibles y se optó por el driver Quimat 3-36V/15A H-Bridge DC, ver Figura 3.17, las características relevantes del driver son:

- Voltaje nominal DC: 3V-36V
- Corriente Nominal: 15A
- Corriente Pico: 30A
- Operación del 0 a 99% del ciclo de trabajo PWM
- Driver dual

Figura 3.17. Driver de los motores

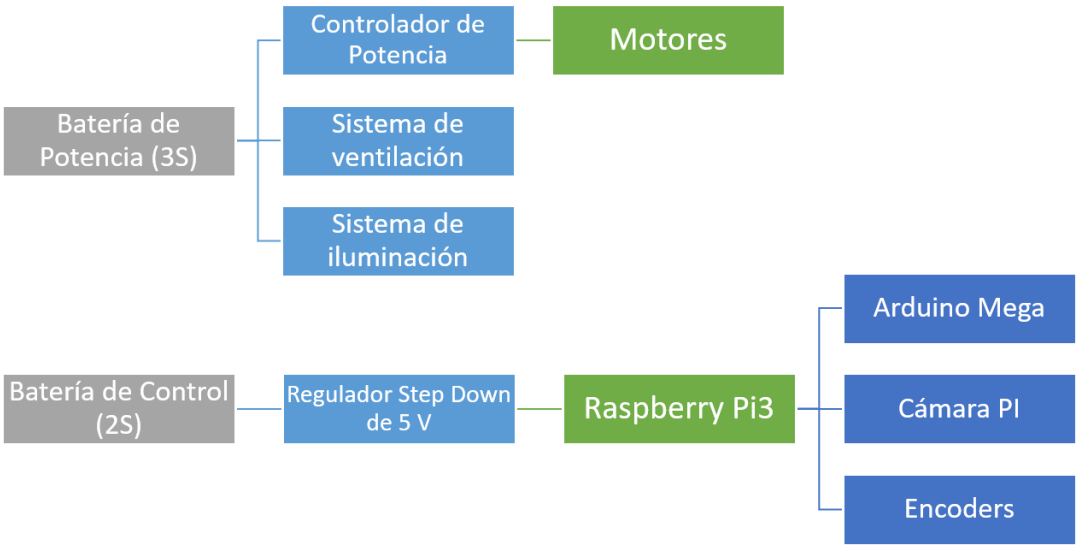
Driver Quimat, Fuente: (Amazon, 2018)

3.5 Sistema de alimentación y autonomía del robot

Para la alimentación del robot se ha utilizado dos baterías de polímero de litio una batería de 7.4v (2S) para el sistema de control y otra batería de 11.1v (3S) para el sistema de potencia, es decir los motores y el driver de los mismos. Como se ha mencionado en el presente capítulo se buscó una cierta autonomía y para ello se realizó un cálculo, con el fin de estimar la autonomía del robot, para ello se procedió a medir la corriente consumida por el sistema de control y el sistema de potencia individualmente, lo cual dio aproximadamente 2.5A y 7A respectivamente.

La capacidad de una batería LiPo se mide en miliamperios hora (mAh), y es básicamente una indicación de la corriente que se puede extraer de la batería en una hora hasta que se descargue. Por lo tanto, se eligió una batería de 5000 mAh con dos celdas (2S) la cual suministrará una autonomía aproximada de dos horas al sistema de control y para el sistema de potencia se eligió una batería de 5000 mAh con tres celdas (3S) que suministrará una autonomía aproximada de 73 minutos. En la Figura 3.18 se puede observar la distribución de alimentación.

Figura 3.18. Sistema de alimentación



Sistema de alimentación, Elaborado por: Jorge Chanco Alvear

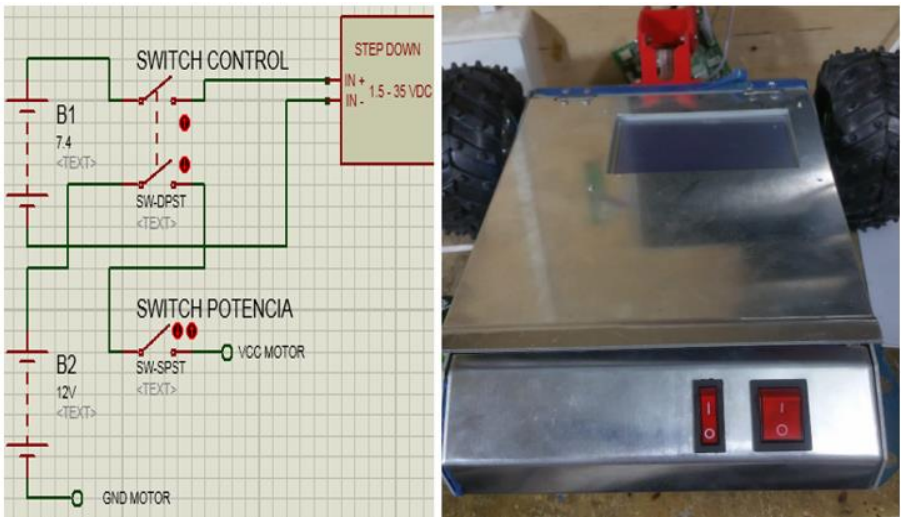
Hay que tomar en cuenta que la tarjeta Raspberry se alimenta con 5v DC, y por lo tanto se usó un regulador de voltaje DC-DC de tipo Step Down de 3A entre la batería de 2S y la tarjeta, para mantener el voltaje que la tarjeta necesita, se optó por la implementación del regulador Step Down con las siguientes características:

- Voltaje de entrada: 3.2V a 40V
- Voltaje de salida: 1.25V a 35V regulable
- Corriente de salida máxima: 3A
- Eficiencia máxima de conversión: 92%
- Rizado: < 30mV
- Frecuencia de conmutación: 65KHz
- Temperatura de funcionamiento: -45º a 85º C

3.6 Sistema de encendido

Con el fin de controlar el paso de la corriente de las baterías a los sistemas del robot, se implementó un par de interruptores montados en el chasis, este par de interruptores proveerán energía al robot de modo condicional, es decir, es necesario que el switch de control este activado para que el driver de los motores pueda ser encendido por medio del switch de potencia. En la Figura 3.19 se puede observar el esquemático del sistema de control y los interruptores montados en el chasis.

Figura 3.19. Sistema de encendido

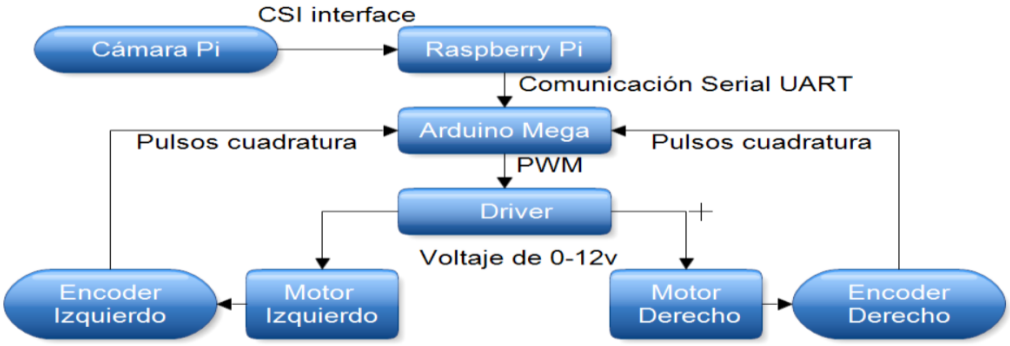


Sistema de encendido, Elaborado por: Jorge Chanco Alvear

3.7 Arquitectura de comunicación entre elementos

En la Figura 3.20 se describe el funcionamiento del robot, en el cual, la cámara envía por medio de la interfaz CSI las imágenes capturadas de la trayectoria a seguir a la Raspberry, la cual procesa la información y se obtiene las coordenadas de la desviación en el eje horizontal, esta información es enviada por medio de comunicación serial UART al Arduino Mega, para que este realice acciones de control sobre el driver de los motores, este driver envía un voltaje entre cero y doce voltios a los motores logrando de esta manera el movimiento del robot, además para conocer el estado de los motores el encoder de cada motor envía pulsos de cuadratura al Arduino Mega para realimentar el sistema de control.

Figura 3.20. Diagrama de comunicación entre elementos



Arquitectura de comunicación entre elementos, Elaborado por: Jorge Chanco Alvear

En la Figura 3.21 se observa el diagrama de flujo de la Raspberry Pi, en la parte inicial se realiza la importación y configuración de las librerías que se utilizan para el procesamiento de imágenes así como la comunicación con la placa Arduino Mega, posteriormente se envía comandos de iniciación para la sincronización del control y se espera la confirmación por parte de Arduino, una vez realizada la confirmación se comienza la captura de imágenes por medio de la cámara. Las imágenes capturadas son convertidas de formato BGR a HSV, los cuales pasan a ser binarizados para el cálculo de bordes, de la totalidad de bordes encontrados se procede a calcular el área que conforman los contornos, la misma que si es igual o mayor a un valor de umbral se procede al cálculo del centroide, este dato muestra la desviación del robot respecto a la trayectoria, este parámetro es enviado a la tarjeta Arduino para el control de motores.

Figura 3.21. Diagrama del algoritmo en Raspberry

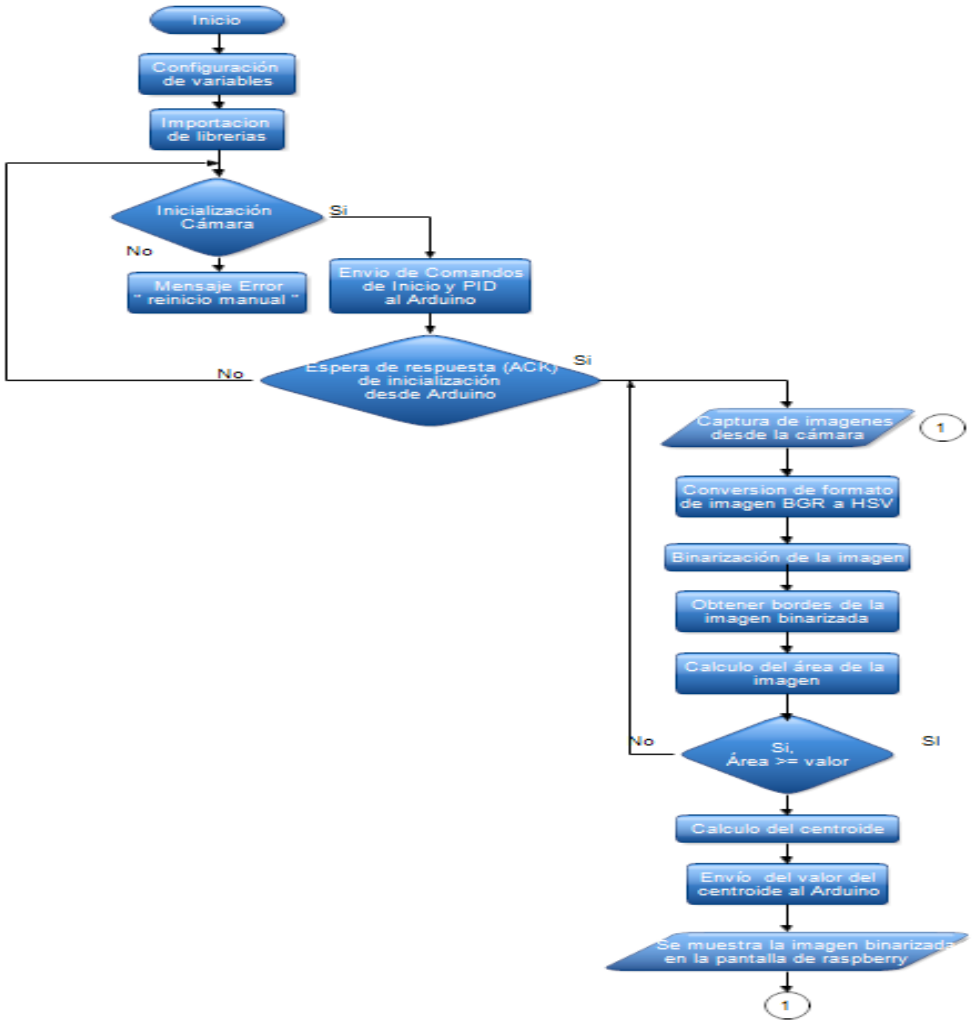


Diagrama del algoritmo en Raspberry, Elaborado por: Jorge Chanco Alvear

En la Figura 3.22, se muestra el diagrama del algoritmo de control PID en el Arduino mega, para inicializar el algoritmo se configuran las constantes del PID, Kp, Ki y Kd, mientras tanto se espera la confirmación de comandos desde raspberry para el inicio del control PID por parte de Arduino, los datos recibidos determinan la desviación del robot con respecto a la trayectoria a seguir, estos parámetros son procesados por el controlador PID, el cual por medio de operaciones matemáticas realiza acciones de control enviando valores de PWM al driver de los motores para corregir dicha desviación.

Figura 3.22. Diagrama del algoritmo PID en Arduino

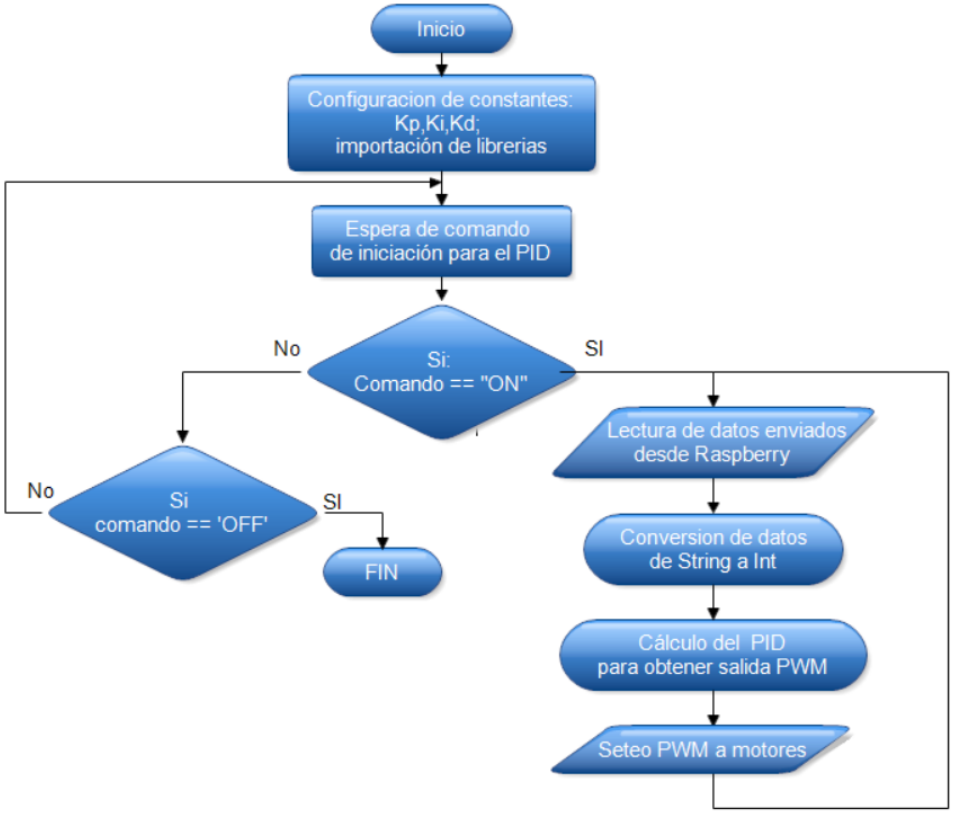


Diagrama del algoritmo PID en Arduino, Elaborado por: Jorge Chanco Alvear

3.8 Controlador PID

Viene de las siglas proporcional, integrativo y derivativo (PID), el cual es un sistema de control de tipo realimentado, el mismo es muy usado en sistemas de control industrial. El funcionamiento se basa en calcular el error entre el valor actual y el valor que se desea ajustar, el control del error se intenta minimizar ajustando la entrada del sistema.

El proporcional P del controlador depende del error actual, el integral I depende de la suma de todos los errores pasados y el derivativo D se basa en errores futuros, mediante la suma de los tres términos se puede ajustar el proceso mediante elementos de control, en el presente proyecto se ajustará la posición del robot en referencia a la línea a ser seguida. En la Figura 3.23 se observa un esquemático simplificado del controlador PID.

Figura 3.23. Controlador PID

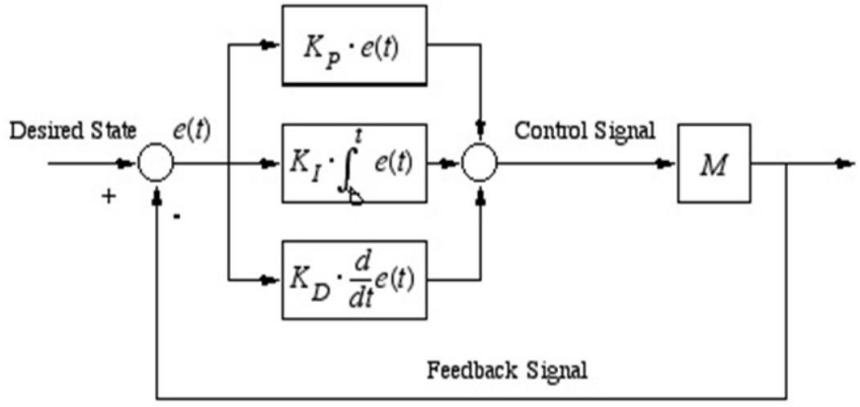


Diagrama de bloques del controlador PID, Fuente: (Lokesh Kumar , Bhavesh Kumar , & Benerjee, 2016)

3.8.1 Calibrando las constantes

Mediante el ajuste de las constantes Kp, Ki y Kd en el algoritmo, el controlador es capaz de proporcionar acciones de control especificas según la necesidad.

Los valores para ajustar el controlador se obtienen analizando el comportamiento del sistema que se desea regular, dichos valores esta relacionados entre sí, entonces si se ajusta para una reacción muy rápida por otro lado se obtendrá una oscilación inevitable, por lo tanto, hay que realizar una calibración o ajuste equilibrada. (Kuo , 1996)

Por otro lado, en los controladores PID, se pueden usar variantes, no siempre se necesitan todas las contantes de control, y se pueden usar solo PI o PD. La ecuación general del PID se observa en la formula Ecuación (3.3).

$$u(t) = k_p * e(t) + k_i * \int_0^t e(t) * dt + k_D * \frac{d}{dt} e(t) \qquad \text{Ecuación (3.3)}$$

De la ecuación se puede tomar en cuenta que:

- Entre más alto sea Kp, más rápida será la respuesta, pero si es muy alto el sistema se hará inestable.
- Los valores altos en Ki eliminan errores estacionarios rápidamente, pero también provoca una sobre oscilación.
- Si Kd aumenta mucho, la sobre oscilación de Ki se reduce, pero aumenta el tiempo de respuesta.

Hay diversos métodos para el ajuste del PID, tal como lo es el método Ziegler-Nichols, pero en este caso se ha utilizado un método heurístico o de prueba y error, basado en la experiencia y las observaciones.

Para obtener la acción proporcional (Kp) se aumentó poco a poco para disminuir el error y aumentar la velocidad de respuesta, pero al ver que el robot se vuelve inestable, se aumenta la acción derivativa (Kd) para conseguir de nuevo la estabilidad. Si el error es muy grande se aumenta la acción integral (Ki) hasta que el error se minimice con la rapidez deseada. Si el sistema se vuelve inestable, se debe aumentar la acción derivativa otra vez. Este método heurístico se basa en conseguir un equilibrio entre las constantes Kp, Ki y Kd según la estabilidad del robot.

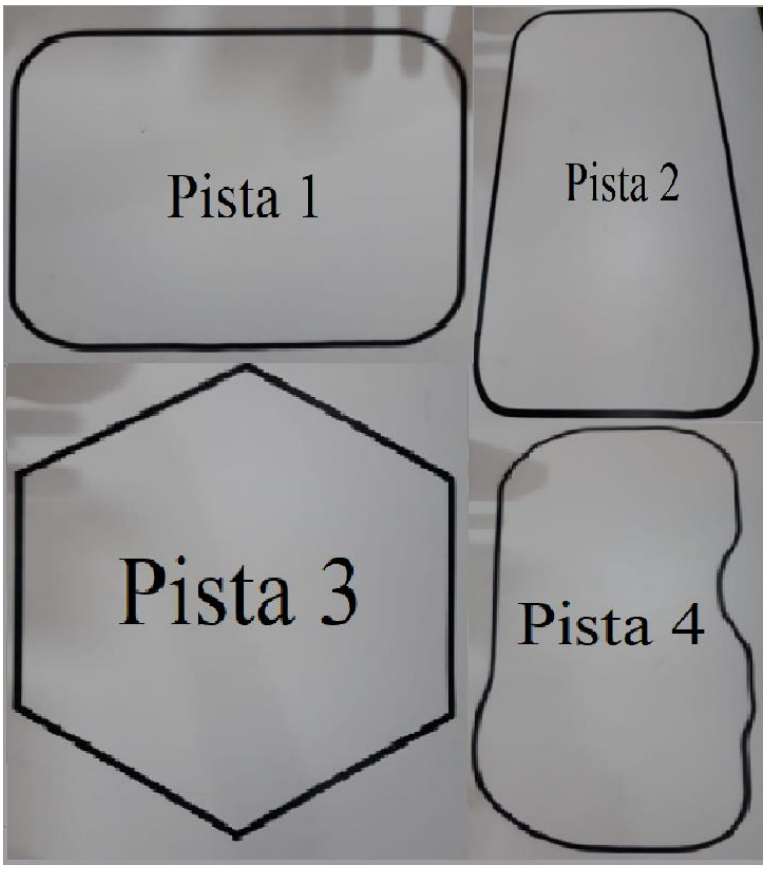
CAPÍTULO 4
PRUEBAS – RESULTADOS

Con el fin de validar el sistema, es decir el hardware y software del robot se realizaron pruebas en pistas planas para observar fallas y ejecutar mejoras al robot. El objetivo del robot en la pista es completar la trayectoria propuesta, y según las veces que logre completar sus objetivos se puede calcular la efectividad de este.

4.1 Pruebas en Pista

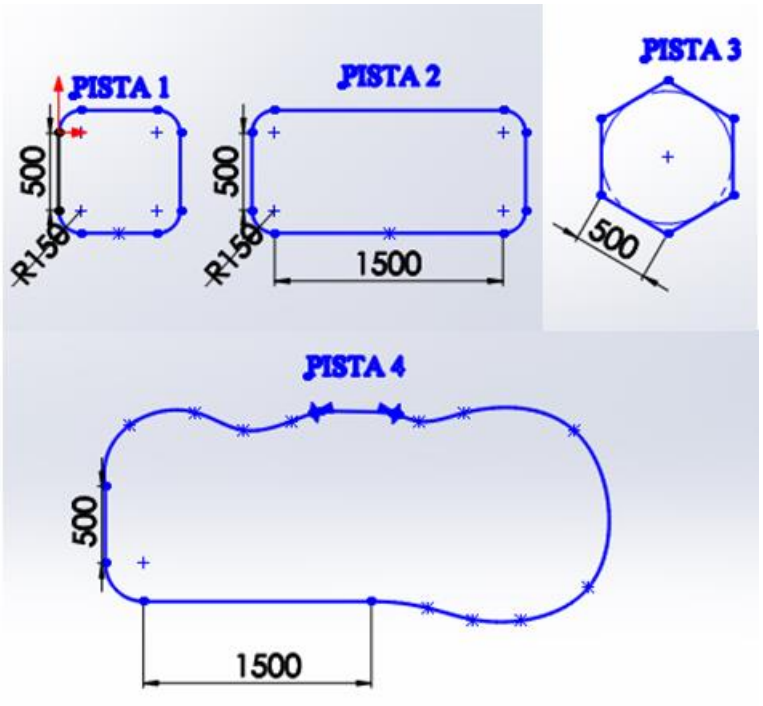
Para comprobar la efectividad del robot se realizaron pruebas en cuatro pistas diferentes que se observan en la Figura 4.1, se tomó diez repeticiones realizadas en sentido horario y otras diez en sentido antihorario. Además, en la Figura 4.2 se observa el diseño y medidas de las pistas.

Figura 4.1. Pistas para pruebas



Detalle de pistas, Elaborado por: Jorge Chanco Alvear

Figura 4.2. diseño de pistas para pruebas



Diseño para las pistas en SolidWorks , Elaborado por: Jorge Chanco Alvear

Tabla 4.1. Resultados en pista 1 sentido horario

PISTA 1 SENTIDO HORARIO (Pista Irregular)			
Prueba #	Tiempo (Seg.)	Perdida de trayectoria	% Trayectoria completada
1	46,2	NO	100
2	46,8	NO	100
3	47	NO	100
4	62,2	NO	100
5	49	NO	100
6	47	NO	100
7	49,5	NO	100
8	61,1	NO	100
9	47,2	NO	100
10	48,3	NO	100
Tiempo promedio	50,43		

Pista de pruebas 1 sentido horario, Elaborado por: Jorge Chanco

De los resultados en la pista 1 en sentido horario de la Tabla 4.1, se determina que no existe perdida en las trayectorias, debido a que el robot responde correctamente a curvas no menores de 15cm de radio, por lo tanto, tiene el 100% de efectividad en terminar el recorrido, con un tiempo promedio de 50.43 segundos.

Tabla 4.2. Resultados en pista 1 sentido antihorario

PISTA 1 SENTIDO ANTIHORARIO			
Prueba #	Tiempo (Seg.)	Perdida de trayectoria	% Trayectoria completada
1	62,2	NO	100
2	49,7	NO	100
3	48,2	NO	100
4	50,1	NO	100
5	49,2	NO	100
6	20	SI	30
7	49	NO	100
8	57	NO	100
9	50	NO	100
10	48	NO	100
Tiempo promedio	51,48		

Pista de pruebas 1 sentido antihorario, Elaborado por: Jorge Chanco

De la Tabla 4.2 se demuestra una efectividad del 90% en terminar el recorrido con un tiempo promedio de 51.48 segundos

Tabla 4.3. Resultados en pista 2 sentido horario

PISTA 2 SENTIDO HORARIO (Pista rectangular con esquinas curvas)			
Prueba #	Tiempo (Seg.)	Perdida de trayectoria	% Trayectoria completada
1	47,06	NO	100
2	49,57	NO	100
3	49,66	NO	100
4	45,52	NO	100
5	47,62	NO	100
6	48,86	NO	100
7	46,96	NO	100
8	46,58	NO	100
9	47,54	NO	100
10	45,99	NO	100
Tiempo promedio	47,536		

Pista de pruebas 2 – sentido horario, Elaborado por: Jorge Chanco

De los resultados en la pista 2 en sentido horario de la Tabla 4.3, se determina un 100% de efectividad en terminar el recorrido, con un tiempo promedio de 47.53 segundos, ya que el robot responde correctamente a curvas no menores de 15cm de radio.

Tabla 4.4. Resultados en pista 2 sentido antihorario

PISTA 2 SENTIDO ANTIHORARIO			
Prueba #	Tiempo (Seg.)	Perdida de trayectoria	% Trayectoria completada
1	51,07	NO	100
2	59,27	NO	100
3	46,65	NO	100
4	49,31	NO	100
5	57,8	NO	100
6	62,8	SI	60
7	46,2	NO	100
8	49,8	NO	100
9	50,1	NO	100
10	51,06	NO	100
Tiempo promedio	52,406		

Pista de pruebas 2 – sentido antihorario, Elaborado por: Jorge Chanco

De los resultados en la pista en sentido anti horario de la Tabla 4.4, se determina un 90% de efectividad en terminar el recorrido, con un tiempo promedio de 47.53 segundos.

Tabla 4.5. Resultados en pista 2 sentido horario

PISTA 3 SENTIDO HORARIO (cuadrado con esquinas curvas)			
Prueba #	Tiempo (Seg.)	Perdida de trayectoria	% Trayectoria completada
1	39,92	NO	100
2	38,99	NO	100
3	39,26	NO	100
4	40,01	NO	100
5	39,56	NO	100
6	41,59	NO	100
7	36,63	NO	100
8	39,04	NO	100
9	39,25	NO	100
10	40,75	NO	100
Tiempo promedio	39,5		

Pista de pruebas 3 – sentido horario, Elaborado por: Jorge Chanco

De los resultados en la Tabla 4.5, se determina un 100% de efectividad en terminar el recorrido, con un tiempo promedio de 39.5 segundos, ya que el robot responde correctamente a curvas no menores de 15cm de radio.

Tabla 4.6. Resultados en pista 2 sentido antihorario

PISTA 3 SENTIDO ANTIHORARIO			
Prueba #	Tiempo (Seg.)	Perdida de trayectoria	% Trayectoria completada
1	47,4	NO	100
2	40,05	NO	100
3	42,29	NO	100
4	38,77	NO	100
5	42,78	NO	100
6	40,59	SI	60
7	41,43	NO	100
8	42,01	NO	100
9	40,26	NO	100
10	41,09	NO	100
Tiempo promedio	37.67		

Pista de pruebas 3 – sentido horario, Elaborado por: Jorge Chanco

En la Tabla 4.6 se observa en tiempo promedio de 37.67, con un 90% de trayectorias completas.

Tabla 4.7. Resultados en pista 4 sentido horario

PISTA 4 SENTIDO HORARIO (Pista Forma de Hexágono)			
Prueba #	Tiempo (Seg.)	Perdida de trayectoria	% Trayectoria completada
1	33	SI	95
2	36,2	NO	100
3	37,01	NO	100
4	35,9	NO	100
5	36,25	NO	100
6	36,28	NO	100
7	36,33	NO	100
8	37,29	NO	100
9	36,26	NO	100
10	35,96	NO	100
Tiempo promedio	36,048		

Pista de pruebas 4 – sentido horario, Elaborado por: Jorge Chanco

En la Tabla 4.7 se observa un 90% de efectividad en trayectorias completadas con un tiempo promedio de 36.04

Tabla 4.8. Resultados en pista 4 sentido antihorario

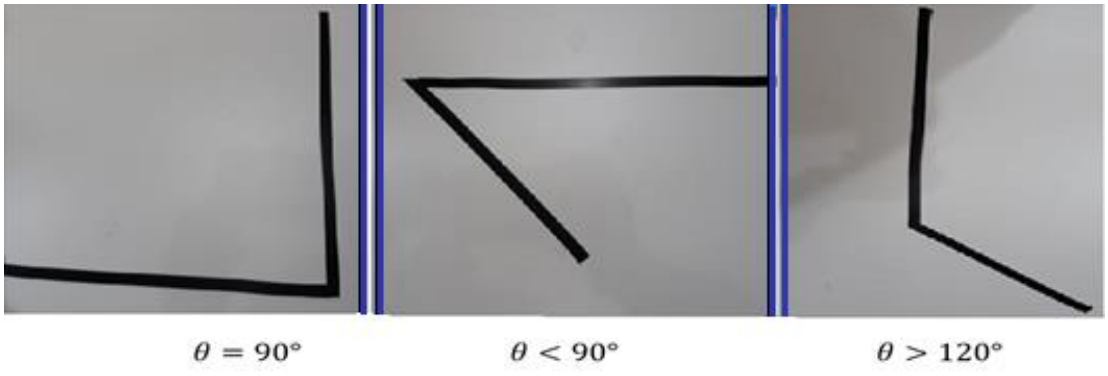
PISTA 4 SENTIDO ANTIHORARIO			
Prueba #	Tiempo (Seg.)	Perdida de trayectoria	% Trayectoria completada
1	34,25	NO	100
2	35,97	NO	100
3	37,25	NO	100
4	38,03	NO	100
5	40,23	NO	100
6	40,3	NO	100
7	38,19	NO	100
8	42,71	NO	100
9	42,7	NO	100
10	41,72	NO	100
Tiempo promedio	39,135		

Pista de pruebas 4 – sentido antihorario, Elaborado por: Jorge Chanco

De los resultados en de la Tabla 4.8, se determina un 100% de efectividad en terminar el recorrido, con un tiempo promedio de 39.13 segundos, ya que el robot responde correctamente a curvas no menores de 15cm de radio.

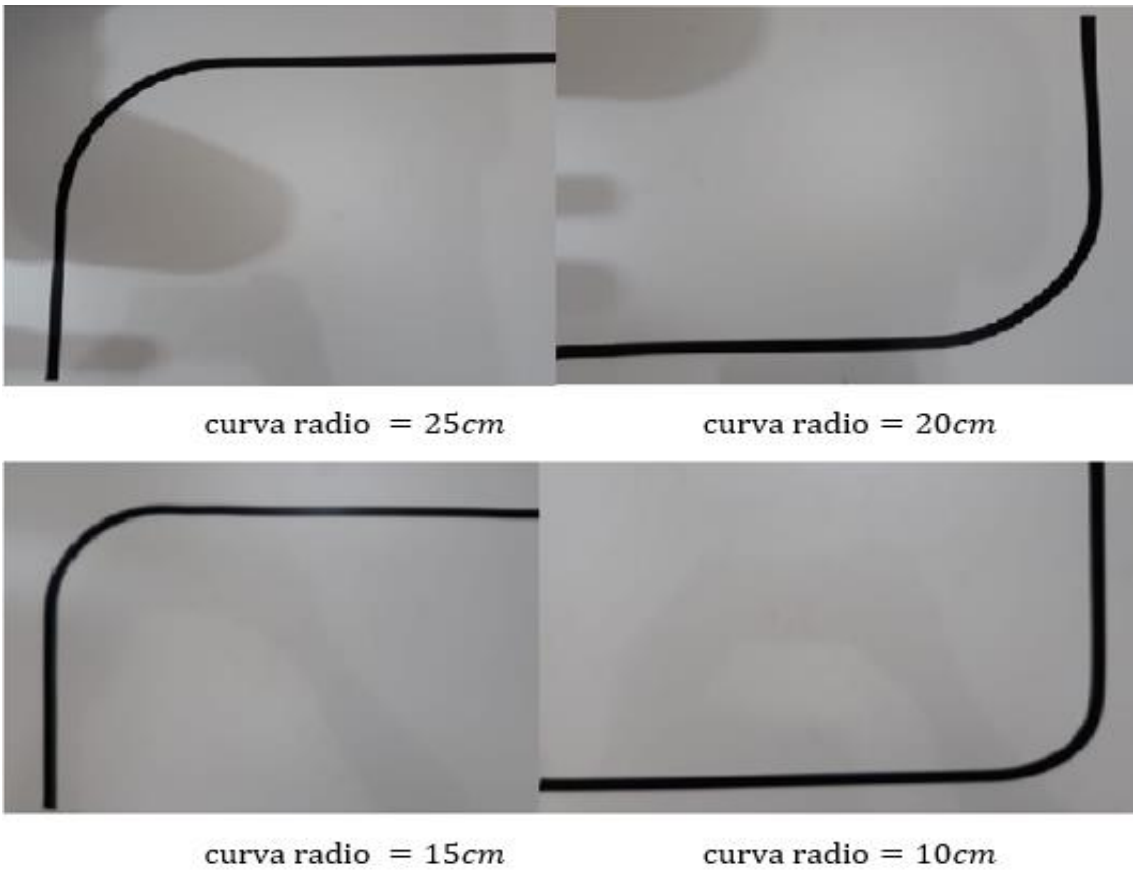
Además, se realizaron pruebas en pista con diferentes ángulos y radios de curvaturas, como se observa en la Figura 4.3 y
Figura 4.4 respectivamente.

Figura 4.3. Curvas con diferentes ángulos



Curvas con diferentes ángulos en pistas, Elaborado por: Jorge Chanco Alvear

Figura 4.4. Curvas con diferentes radios



Curvas con diferentes radios en pistas, Elaborado por: Jorge Chanco Alvear

De las pruebas realizadas en pistas verifico que el robot es eficiente en curvas cuyo radio es mayor o igual a 10 cm y trayectorias con ángulos de apertura mayores a 90 grados.

CONCLUSIONES

- En este trabajo de titulación se logró implementar un algoritmo de visión artificial el cual permite al robot diferencial seguir una línea negra descrita sobre una superficie blanca plana; el proceso se validó con pruebas de campo por medio de cuatro pistas cerradas con diversos grados de complejidad. Las pruebas fueron en sentido horario y antihorario, completando la trayectoria en su totalidad en cada una de las mismas, obteniéndose como resultados un 90% de efectividad de ejecución en cada prueba, lo que significa que la visión artificial controla la navegación del robot.

- Se identifico al momento del desarrollo del marco teórico que existen varios algoritmos de visión artificial aplicados al seguimiento de una línea de los cuales se implementaron dos de ellos; los mejores resultados se obtuvieron con el algoritmo de conversión de imagen de formato BGR a HSV, el cual permite al robot navegar con mayor rapidez reduciendo el tiempo de latencia generado por el procesamiento de imágenes.

- Para la selección de la tarjeta controladora procesadora de la visión artificial se analizaron varias posibilidades, determinando que la placa de desarrollo Raspberry Pi 3 es adecuada para la implementación del algoritmo de visón artificial; ya que, permite ejecutar los algoritmos con mucha fluidez permitiendo al robot procesar los datos y lograr culminar las trayectorias propuestas en las pruebas sin dificultad.

- En la implementación del robot diferencial se demostró que este mecanismo es adecuado para el prototipo propuesto, ya que cumple con los requerimientos planteados de portabilidad, autonomía, protección y movilidad; el cual integra todos los elementos requeridos para el eficiente funcionamiento del robot dotando al mismo de estabilidad y robustez estructural.

- Con las pruebas de campo se validó el funcionamiento del sistema en cuatro pistas con diferentes grados de complejidad, las cuales permitieron observar los errores tanto de hardware como de software en el prototipo; de esta manera realizar correcciones a los dos componentes y lograr que las pistas propuestas sean concluidas en su totalidad por el robot; de tal forma, que esto demuestra los índices de efectividad los cuales están comprendidos entre el 90% y 100%.

RECOMENDACIONES

- Para mantener un adecuado nivel de voltaje de operación de las baterías, se recomienda la implementación de un sistema de control de nivel de carga, ya que el bajo nivel de voltaje en las baterías afecta directamente al desempeño del robot reduciendo le velocidad de movimiento y afectando al sistema de control además un nivel de voltaje por debajo del nominal reduce la vida útil de las mismas por ende un gasto económico adicional al prototipo.

- Con el fin de garantizar un movimiento del prototipo libre de atascos, se recomienda la implementación de una rueda loca construida con base de plástico el mismo que posee un coeficiente de rozamiento menor que el de la bola loca con soporte metálico.

- Para concluir con éxito una trayectoria la cual integre líneas con ángulos menores o iguales a 90°, se recomienda la implementación de un algoritmo el cual permita identificar patrones o señales que indique un cambio en la trayectoria o muestre un cruce entre líneas, para impedir que el prototipo pierda la línea a seguir.

- Para dotar de un nivel mayor de autonomía, se recomienda la implementación de un algoritmo que detecte colores o formas que determinen la finalización de una trayectoria o ejecutar acciones específicas según el color o la forma detectada.

REFERENCIAS BIBLIOGRÁFICAS

Aguilera Hernández, M., Bautista, M., & Iruegas, J. (2007). Diseño y Control de Robots Móviles. Nuevo Laredo, México. Obtenido de http://www.mecamex.net/anterior/cong02/papers/art24.pdf

Amazon. (25 de Junio de 2018). *Amazon*. Obtenido de https://www.amazon.com/Driver-Arduino-Quimat-H-Bridge-Circuit/dp/B06X96MNQC

Arévalo, V., González , J., & Ambrosio, G. (2002). LA LIBRERÍA DE VISIÓN ARTIFICIAL OPENCV. 6.

ATMEL. (Noviembre de 2015). *ATMEL 8-BIT microcontroller with 4/8/16/32 kbytes in system programmable flash*. Recuperado el 10 de Diciembre de 2016, de http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

Balajia, V., Balajib, M., Chandrasekaran, M., Ahamed khan, M., & Elamvazuthie, I. (2015). *Optimization of PID Control for High Speed Line Tracking Robots.* Recuperado el 2017, de ScienceDirect: www.sciencedirect.com

Beng Kiat, N. (2009). *uMouse lecture.* Recuperado el 10 de Agosto de 2016, de https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxuZ2JlbmdraWF0ZG9jfGd4OjM0YzM0Y2EyYThmNjI5MTc

Bradski , G., & Kaehler, A. (2008). *Learning OpenCV*. CA: O'Reilly Media.

Cárdenas Calle, N. R., & Simbaña Quinde, G. R. (2007). *Diseño y construcción de un robot movil didactico para estudios experimentales de robotica cooperativa.* Cuenca: Universidad Politécnica Salesiana.

Castro, A. (23 de Enero de 2014). *¿Qué es Raspberry Pi, dónde comprarla y cómo usarla?* Obtenido de ComputerHoy.com: https://computerhoy.com/noticias/hardware/que-es-raspberry-pi-donde-comprarla-como-usarla-8614

Ferro Laspidea, P., & Goicoechea Fernández, J. (2014). *Diseño e implementación de un robot para la automatizacion de un almacén.* Pamplona: Universidad pública de Navarra.

García Villanueva, M., Ramírez Zavala, S., & Ortega Reyes, H. (2016). *Robot Móvil Autónomo Seguidor de Línea con Raspberry Pi y Sistema de Visión Artificial.*

Gobierno de España, Ministerio de educacion. (2012). *Aplicación práctica de la visión artificial en el control de procesos industriales. Visión Artificial.* España.

Gonzáles Marcos, A., Martínez de Pisón Ascacíbar, F., Pernía Espinoza, A. V., Alba Elías, F., Castejón Limas, M., & Ordieres Meré, J. (2006). *Técnicas y algoritmos básicos de visión artificial.*

Griffith, B. (2 de Enero de 2017). *Las mejores diez placas de desarrollo de 2017*. Obtenido de https://www.arrow.com/es-mx/research-and-events/articles/the-top-ten-development-platforms-dev-kits-for-2017

Heath , N. (30 de Noviembre de 2017). *What is the Raspberry Pi 3? Everything you need to know about the tiny, low-cost computer*. Obtenido de https://www.zdnet.com/article/what-is-the-raspberry-pi-3-everything-you-need-to-know-about-the-tiny-low-cost-computer/

Hernández Millán, G., Ríos González, L. H., & Parra Lara, H. (Abril de 2012). *Implementacion de un controlador PID mediante RNA para el control de motores D.C de robots moviles diferenciales.* Obtenido de http://revistas.utp.edu.co/index.php/revistaciencia/article/view/6669

Holland, J. M. (2004). *Designing Autonomous Mobile Robots Inside the Mind of an Intelligent Machine.* British: Elsevier.

Huamán , A. (2018). *OpenCV tutorial, Canny Edge detector.*

INCIBE. (08 de febrero de 2018). *Introducción a los sistemas embebidos*. Obtenido de https://www.certsi.es/blog/introduccion-los-sistemas-embebidos

Kenneth Dawson, H. (2014). *A practical introduction to Computer visión with OpenCV.*

Kuo , B. (1996). *Sistemas automáticos de control.* España: Prentice Hall.

Lokesh Kumar , A., Bhavesh Kumar , C., & Benerjee, G. K. (2016). A Review on Hybrid Controller Using Soft Computing Algorithms. *American Journal of Electrical and Electronic Engineering*, 49-61.

Neal, A. (enero de 2010). *Tips for selecting DC motors for your mobile robot.* Recuperado el Diciembre de 2016, de Servomagazine: http://www.servomagazine.com/uploads/issue_downloads/pdf/Tips%20For%20Selecting%20DC%20Motors%20For%20Your%20Mobile%20Robot.pdf.

Pololu Robotics and Electronics. (23 de junio de 2018). *Pololu*. Obtenido de https://www.pololu.com/product/3217

Pololu Robotics and Electronics. (24 de junio de 2018). *Pololu*. Obtenido de https://www.pololu.com/product/1557

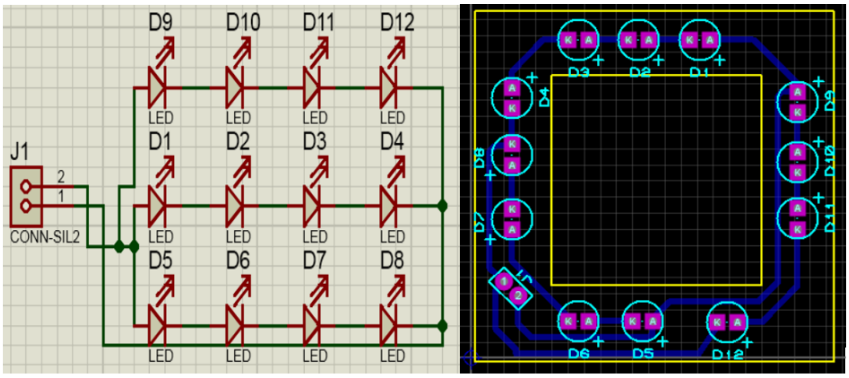
RASPBERRY PI FOUNDATION. (s.f.). *Raspberry Pi org.* Obtenido de https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Romo Caicedo, D. A., & Toca Trujillo, L. E. (2016). *Diseño e implementación de dos robots de competencia* . Quito: Escuela Politécnica Nacional.

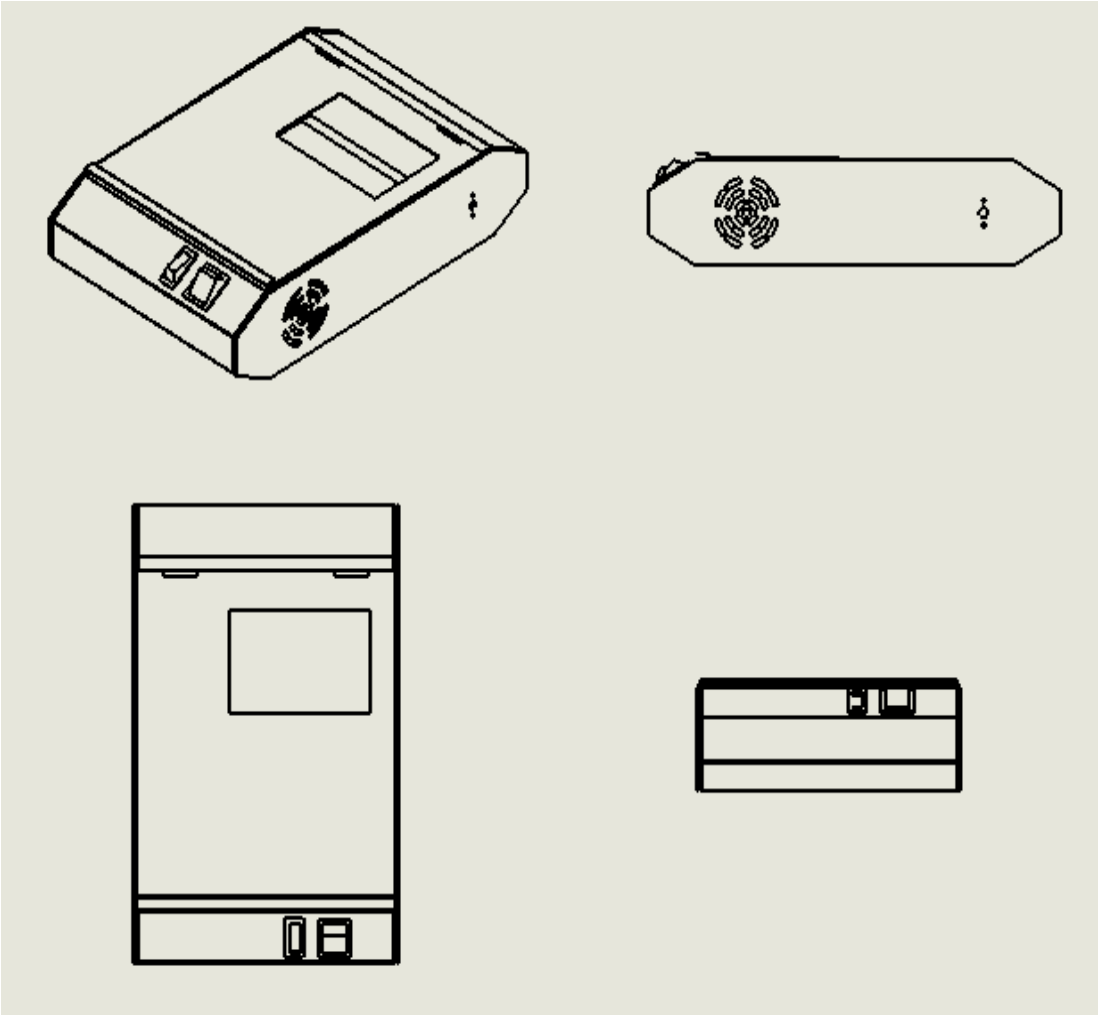
Vinay, R. (6 de Julio de 2017). *Developer Zone*. Obtenido de https://software.intel.com/en-us/articles/what-is-opencv

ANEXOS

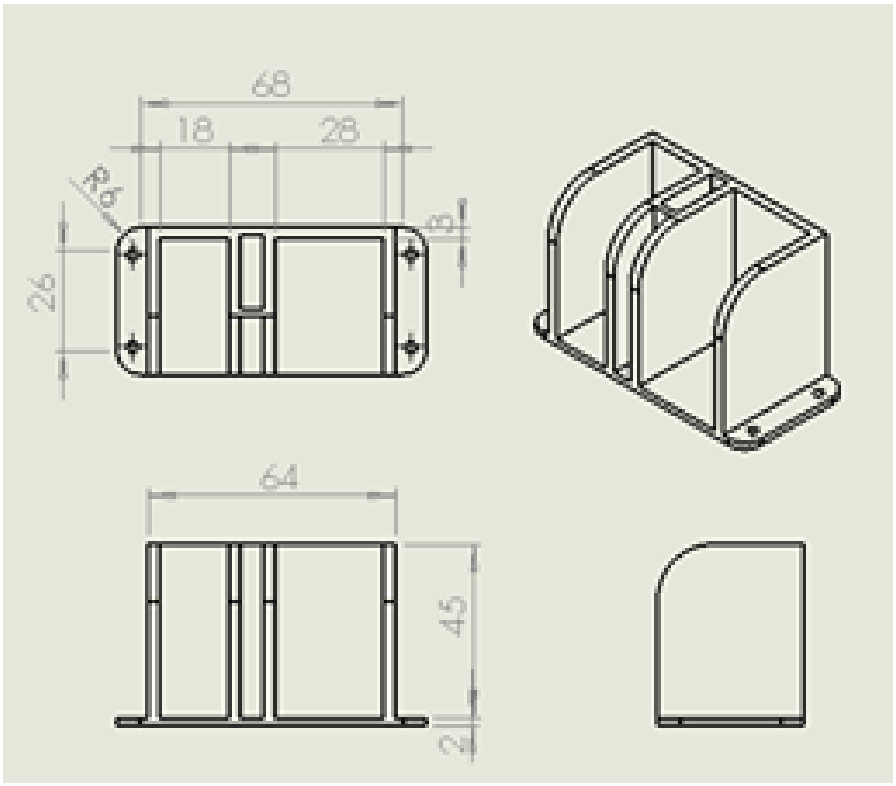
Anexo 1. Circuito y PCB de luces para cámara Pi



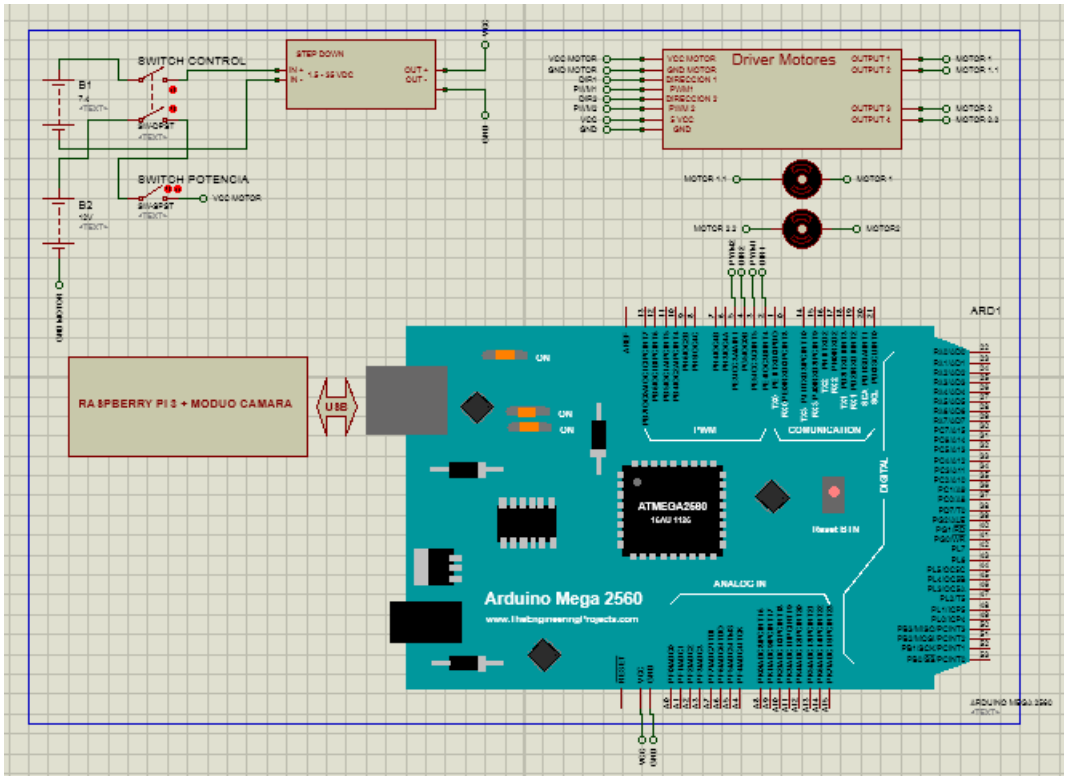
Anexo 2. Planos del chasis del robot, vistas en general



Anexo 3. Planos de los soportes para baterías, vistas en general



Anexo 4. Esquemático general de sistema electrónico del control del robot



Anexo 5. Esquemático del sistema de encendido del robot